CONTROL DATA® 6600 Computer System Operating System/Reference Manual

SIPROS 66

SImultaneous PRocessing Operating System

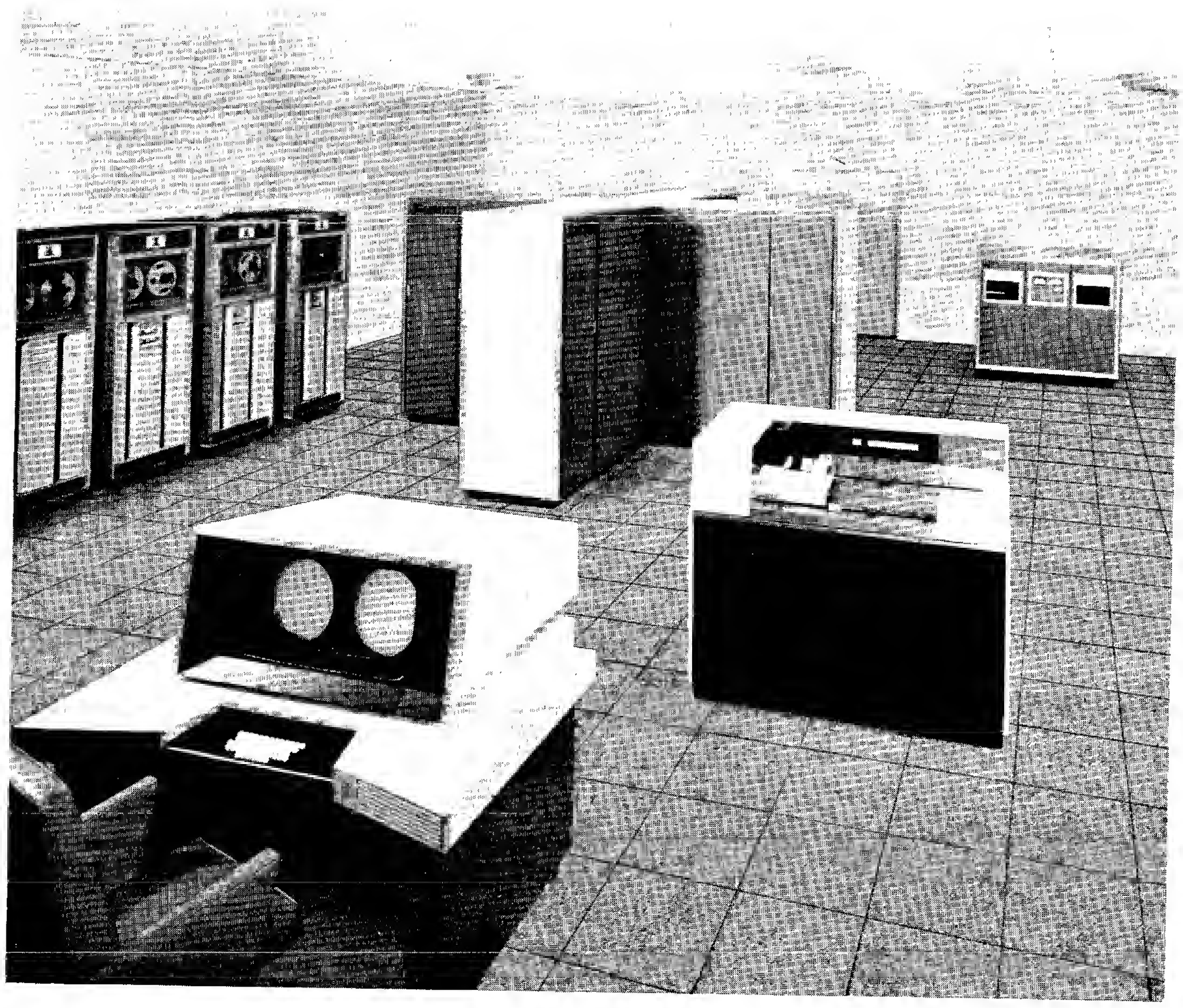
FIRST EDITION

PREFACE

This manual describes the capabilities of the Control Data® 6600 Operating System – SIPROS 66. Detailed information on the system macros, information concerning the processing flow, principles of operation, and features available to the programmer are included.

The information provided in this manual, together with the information in the 6600 Programming System publications (the FORTRAN, ASCENT, and ASPER publications), is sufficient to allow the writing of operational programs.

TABLE OF CONTENTS



6600 COMPUTING SYSTEM

Main frame *(center)—contains 10 peripheral and control processors, central processor, central memory, some I/O synchronizers.*

Display console *(foreground)—includes a keyboard for manual input and operator control, and two 10-inch display tubes for display of problem status and operator directives.*

CONTROL DATA 607 tapes *(left front)— ½ inch magnetic tape units for supplementary storage; binary or BCD data handled at 200, 556, or 800 bpi.*

CONTROL DATA 626 tapes *(left rear)—1-inch magnetic tape units for supplementary storage; binary data handled at 800 bpi.*

Disc file *(right rear)—Supplementary mass storage device holds 500 million bits of information.*

CONTROL DATA 405 card reader *(right front)—reads binary or BCD cards at 1200 card per minute rate.*

INTRODUCTION

The Control Data® 6600 Computing System is a complex of computers and peripheral equipment with unprecedented operating modes and capabilities for multiprocessing. It has been designed to handle special applications in which on-line inputs and outputs with real-time calculations are involved as well as large volumes of conventional applications. In addition, it has been designed to handle multiple problems of both types, real-time and conventional, simultaneously. Efficient application of this hardware to several jobs simultaneously in a dynamic job environment is one of the basic objectives of the SImultaneous PRocessing Operating System, SIPROS 66.

Wherever possible, SIPROS 66 has been designed as a parametric system. Where operations of the system require buffer regions, the lengths are controlled with parameters; where the operations require peripheral devices, the units and number of units are designated by parameters; and where time and utilization limits are enforced, the actual values of the limits are parameters. Through selection of parameter values, it is expected that the multiprocessing properties of the system can be balanced to the average job mix of a user. In this way, SIPROS 66 can be made to fit the individual user's needs without the cost of system modification.

SYSTEM ORGANIZATION

The CONTROL DATA® 6600 is a large-scale, solid-state, general-purpose digital computing system. The advanced design techniques incorporated in the system provide for extremely fast solutions to data processing, scientific and control center problems.

Within the 6600 are eleven independent computers (Fig. 1). Ten of these are constructed with the peripheral and operating system in mind. These ten have separate memory and can execute programs independently of each other or the central processor. The eleventh computer, the central processor, is a very high-speed arithmetic device. The common element between these computers is the large central memory.

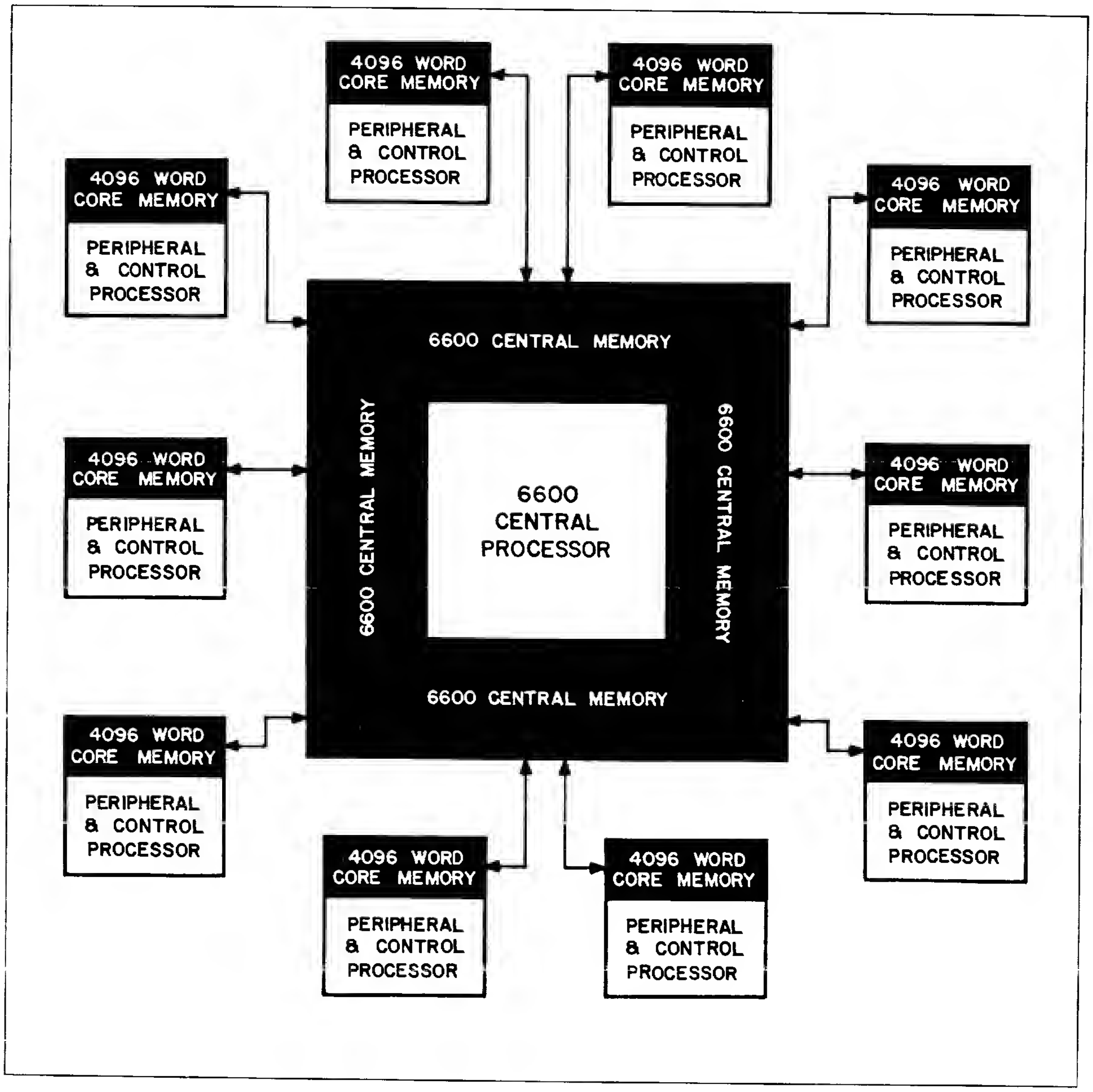
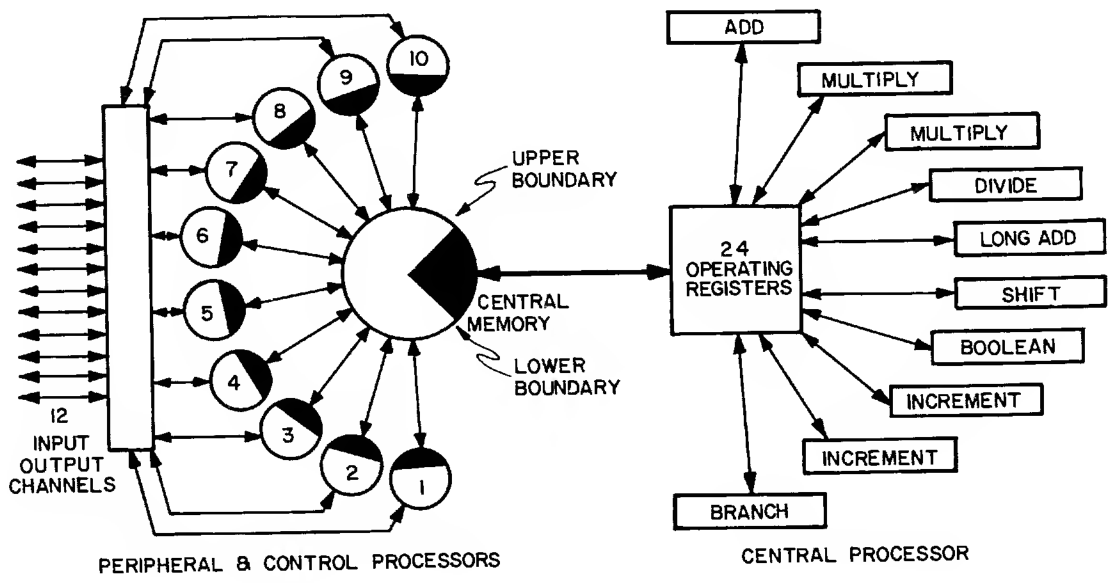


Figure 1 CONTROL DATA 6600



CENTRAL MEMORY

– 131,072 words

– 60-bit words

– Memory organized in 32 logically independent banks of 4096 words with corresponding multi-phasing of banks

– Random access, coincident-current, magnetic core

– One major cycle for read-write

– Maximum memory reference rate to all banks — one address/minor cycle

– Maximum rate of data flow to/from memory — one word/minor cycle

DISPLAY CONSOLE

– Two display tubes

– Modes
Character
Dot

– Character size
Large — 16 characters/line
Medium — 32 characters/line
Small — 64 characters/line

– Characters
26 alphabetic
10 numeric
11 special

Figure 2 BLOCK DIAGRAM OF 6600

1. SIPROS OPERATION

During normal operation, SIPROS 66 assumes full control of the system and multiprocesses jobs to the extent of job and equipment availability. If an operator intervenes or a high priority job is encountered, SIPROS 66 continues to maintain a flow of jobs as input into a disk stack. It schedules jobs from the stack based upon priorities, equipment, and memory availability, keeps utilization accounts, catalogs output data from the jobs in execution, and schedules output data to peripheral devices. This procedure is described more thoroughly below in terms of the tasks performed by the various parts of the system.

1.1 SYSTEM CONFIGURATION

SIPROS 66 is a highly flexible system which has been designed to operate with many different hardware configurations. A basic system is assumed, however, since SIPROS 66 is made up of several programs which reside and operate in various components of the computer system. (A summary of hardware characteristics is given in Appendix 1.) The assumed configuration consists of a main frame containing a central processor with 131,072 60-bit words of magnetic core memory, ten peripheral processors with 4096 12-bit words of magnetic core memory each, and, as a minimum, one each of the following:

- Disk unit with 37,355,520 12-bit words
- Display console
- 1200 card/minute reader
- 1000 line/minute printer
- 250 card/minute punch
- Bank of two 607 or 626 magnetic tape units

1.2 EQUIPMENT UTILIZATION

This section outlines the utilization of the various components of the Control Data® 6600 Computer. The functional groups are as follows:

- Central Processor and Memory
- System Disk
- Peripheral Processors
- Other Equipment

Figure 3 illustrates these groups and the intercommunication required by SIPROS 66.

1.2.1 CENTRAL PROCESSOR AND MEMORY

The central processor and memory operate under the control of the Executive which is housed in one of the peripheral processors. The primary function of the central processor is handling the computational load as directed by the Executive. This includes operating on data provided for production runs, and compiling or assembling new programs written in the various programming languages. The primary function of central memory is storing operational and system programs together with the data they require. Central memory also serves as a link between the central processor and peripheral processors since both types of processor may access it.

The operational programs that are stored in central memory are user programs which are to be executed, assembled, or compiled. They are stored in the job area along with data for the programs to be executed and a job table which is used by the Executive to control the scheduling and processing of jobs.

A SIPROS 66 program, the Central Processor Resident, is also stored in the job area of central memory along with each operational program. It interprets system macros, transfers parameters supplied by the macros to the first twelve locations of the operational program, and fills buffer areas with print, punch, and card reader data being transferred to the disk.

A variety of control information required by SIPROS 66 is stored in central memory. It includes: a peripheral library directory; a central library directory; a disk table directory which identifies the physical records on the disk that are in use; a disk release table which tells what disk space to release; disk control information which allows the processors to communicate with one another; and equipment request tables to which peripheral processors look for jobs.

Some of the SIPROS 66 system routines may also be stored in central memory on an optional basis. Included in this group are: loader routines which load jobs onto disk or into central memory; a routine that displays information on the console; overlays

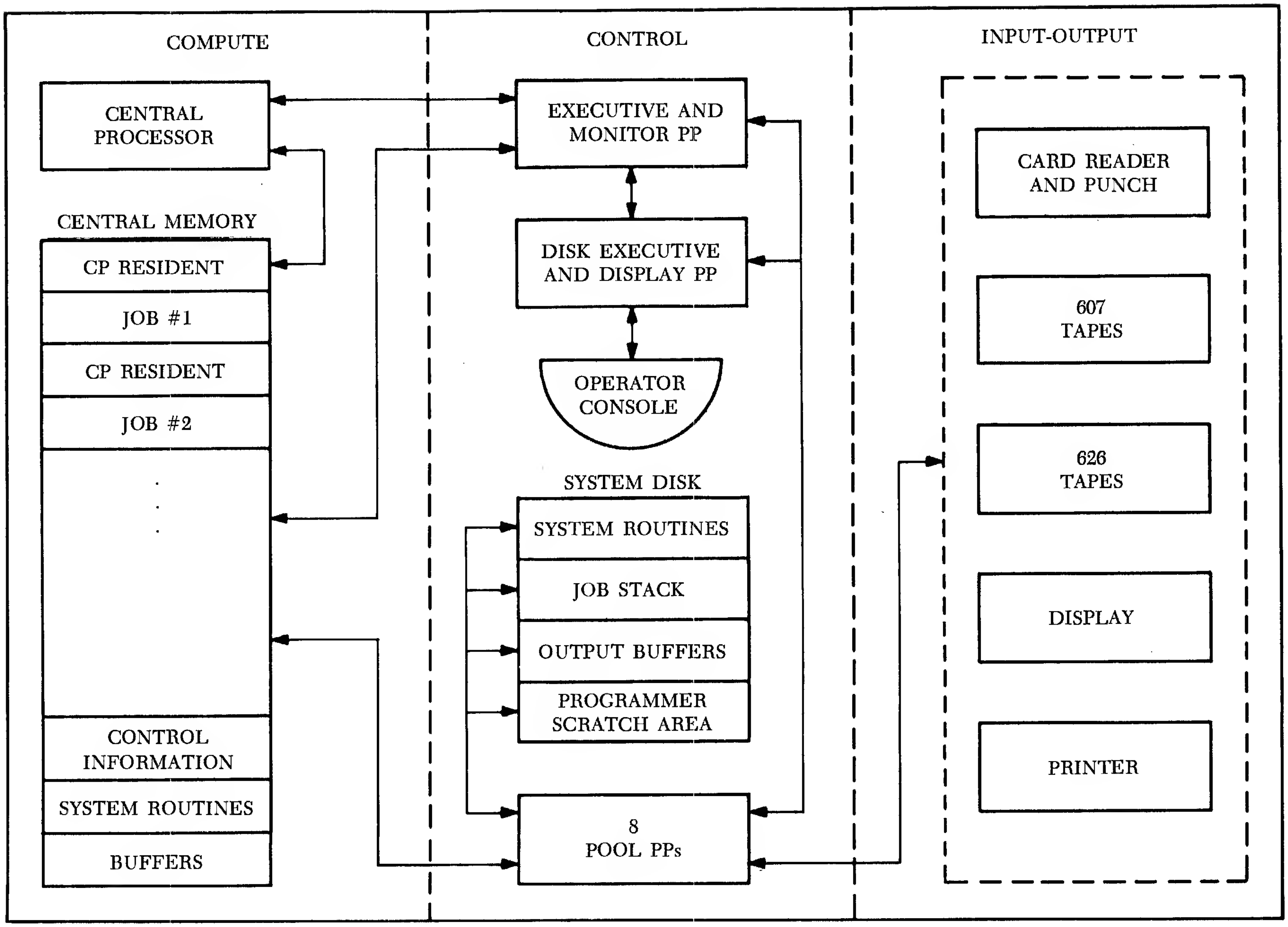


Figure 3. SIPROS COMMUNICATION FLOW

for the SIPROS 66 routines that are in permanent residence in peripheral processors; and the print, punch, and tape packages. The installation decides whether or not central memory space will be used for these routines and, if so, the amount of space that will be used.

Information being transferred by SIPROS 66 system routines to and from the disk is buffered through central memory.

1.2.2 SYSTEM DISK

The system disk unit is logically partitioned into sections as follows:

System Library

Job Stack (Programs and Input Data)

Printer and Card Output Data

Temporary Storage for Object Programs

The system library section provides permanent storage for the programs of SIPROS 66, the programming systems (FORTRAN, ASCENT, and ASPER), and the object code subroutine library. These programs are called into central memory or peripheral processor memory when needed.

The job stack provides temporary storage for source and object programs and data en route to memory. To the extent of job availability, the job stack is dynamically maintained at capacity to provide the Executive with a wide choice in scheduling.

Maintenance of the proper output sequence of data from each job under execution, and utilization of peripheral equipment at full speed, requires temporary disk storage of printer and punch data from each job until the entire job has been completed. Once the task has been completed, the transfer of this data to the appropriate peripheral device is performed in serial and continuous fashion as an off-line operation.

The remainder of the system disk unit is available for temporary use by object programs. SIPROS 66 provides space assignment, information access, and job accounting services for the various programs that are using the disk at any given time. Allocation is based on variable length logical records. These logical records are defined and referenced symbolically through the programming languages and interpreted by SIPROS 66 as combinations and/or parts of fixed length physical records on the disk but are not in any way restricted to fixed length reeords. Fixed length physical records are defined by the installation and may be from eight to 64 sectors in length (as long as each record is a multiple of eight sectors, i.e. a multiple of 512 60-bit words).

1.2.3 PERIPHERAL PROCESSORS

Peripheral processors are used by SIPROS 66 to control and monitor the system and to perform various I/O functions in support of operational and system programs. Two of the peripheral processors are permanently assigned system functions. They are the Executive and Monitor PP and the Disk Executive and Display PP (see Figure 3). The remaining eight processors are assigned tasks on a temporary basis as the need arises.

Executive and Monitor PP

The Executive and Monitor PP is permanently assigned the duties of activities director and operations monitor. The two separate but interrelated programs, the Executive and Monitor, which perform these duties are described in the following paragraphs.

The Executive program has complete charge, from a scheduling standpoint, of the central processor, other peripheral processors, input/output channels and peripheral equipment. It also maintains a status list on each job in the system. The Monitor assists the Executive by continuously watching the progress of the job in central memory upon which the central processor is operating. Any need for action, such as an I/O request, is passed along for appropriate action by the Executive. If the request indicates that no further processing can be accomplished until the request is satisfied (normal, nonbuffered operation), the Executive reassigns the central processor. This causes the Monitor to switch to a new job.

In scheduling the activity of the central processor, the Executive sets up all jobs in central memory in a priority sequence and prepares a list for the Monitor. As the Monitor cycles through the list of jobs and finds wait conditions, it exchange jumps the next job in the priority sequence. On the next iteration, the Executive reschedules the jobs in central memory and prepares a new list for the Monitor. Thus, Executive scheduling of the central processor is based on priorities with I/O waits.

To schedule the activity of other peripheral processors, the Executive arranges the operations to be performed into classes (such as tape operations, print operations, etc.) and assigns processors to the various classes. The number of processors assigned to each class is limited by the installation and is determined by priority and the volume of operations in the class. Upon completion of one operation, the proeessor automatically goes to the next operation in the same class unless instructed otherwise by the Executive. By using this approach, SIPROS 66 avoids the frequent reloading of peripheral processor programs.

Executive scheduling of I/O channels and peripheral equipment is done at the time a job is loaded from the disk into central memory. All jobs in the disk job stack are continuously examined by the Executive. As soon as a job has the proper priority and there is sufficient peripheral equipment available, the Executive builds a load request and sends it to a SIPROS 66 routine ealled the Job Loader. The Job Loader transfers the job from the disk to central memory. During the loading process, the Executive assigns equipment to the job.

The status of each job in the system is maintained by the Executive to provide information telling where the job is and what is happening to it. The Executive continuously examines this status for a change and takes action based on the change. For example, when a job is terminated, the Monitor examines the P register to determine whether there has been a normal or error halt and sets job status to the stop condition. The Executive then calls in the Job Termination Package which builds the print and punch requests and performs a post-mortem dump if one was requested. After the Job Termination Package has completed its functions, the Executive releases the central memory space used by the job. Disk space is not released until the print and punch operations have been completed.

Disk Executive and Display PP

The Disk Executive is the SIPROS 66 program that directs the activities of the system disk. It schedules requests received from the various programs that are using the disk for two slave processors which do the reading and writing. In performing this scheduling function, the Disk Executive searches its wait stack for all requests that can be processed at the current position of the disk read-write heads. After it has identified the ones that can be processed, it sorts them into a sequence that will allow as many of them as possible to be serviced during each revolution of the disk. When this has been done, the Disk Executive instructs the slave processors to read or write. Repositioning of the read-write heads takes place only after all requests for the current position have been processed. If there are multiple system disks in use, the Disk Executive will overlap disk functions by repositioning the read-write heads of one disk while the slaves are reading or writing another disk.

The Display program is used primarily to inform the operator of the status of jobs in central memory. It is also used to give the operator information about tape mounting and to allow him to communicate with the system. In communicating with the system, the operator may display or change priorities, display jobs in the job stack or in central memory, or add a utility routine such as a core dump to a job already in the system.

Pool PPs

The remaining peripheral processors, called Pool PPs, are the task performers. They carry out, under the direction of the Executive, the operations requested by central processor programs or by programs in other peripheral processors. Each time a request is made, the first available Pool PP is assigned the task. When the task has been completed, the Pool PP reports back to the requesting program and to the Executive that it is available for another task.

Each of the Pool PPs has a SIPROS 66 program in permanent resident. This program, the PP Resident, interprets requests sent to it, loads the required system or user program into the peripheral processor's memory, and transfers to that program.

After the system or user program completes its processing, it transfers back to the PP Resident. The PP Resident then notifies the Executive that the request has been completed.

A variety of tasks may be assigned to Pool PPs. In general, these tasks can be broken down into three categories:

System Operations

Off-line Operations

Internal Operations

System operations are initiated by the Executive and always involve central memory or the disk. Some of the common system operations include: disk reading and writing, job loading, and job termination.

Whenever there are disk read or write requests in the disk request stack, two Pool PPs are assigned to work with the Disk Executive as slaves. These slave processors do the reading and writing as directed by the Disk Executive. One slave reads or writes a disk sector (320 12-bit words) while the other transfers a sector to or from central memory. This two-processor approach to disk reading and writing maximizes the disk transfer rate. Although the two slave processors are returned to the pool when not in use, they are normally assigned to the task of disk reading and writing.

Jobs are loaded from cards or tape into the disk job stack by a SIPROS 66 routine called the Batch Loader. This routine is brought in by the Executive which examines the disk job stack to determine whether or not there is space for additional jobs. If there is space and there are jobs to be loaded, the Executive initiates a Batch Loader request and sends it to the resident in a Pool PP. The PP Resident loads the Batch Loader into the peripheral processor's memory and transfers to the loader. The Batch Loader then reads jobs in from cards or tape and transfers them to the disk job stack. It continues to load jobs into the stack as long as there is space in the stack and the Executive continues to build Batch Loader requests. During the loading process, the Batch Loader also constructs a job table entry for each job; this entry is used by the Executive to control the scheduling and processing of the job.

Jobs are loaded from the disk job stack into central memory on a priority basis whenever there is sufficient central memory space and peripheral equipment available. The SIPROS 66 routine that performs this loading function is the Job Loader. This routine is loaded into a Pool PP after the Executive has initiated a request and sent it to that processor. When control has been transferred from the PP Resident to the Job Loader, the Job Loader transfers the job to central memory and reports back to the Executive through the status list that the job is waiting in central memory.

Upon completion of the processing of a job, the Executive sends a request to a Pool PP to load the Job Termination Package. When the Job Termination program has been loaded and receives control, it starts closing out the job. This consists of preparing accounting information, building a dump request if necessary, and building print and punch requests. After a job has been terminated, it is removed from the system.

Other system operations include:

1. Disk-to-printer transfer of printer data stored on the disk,
2. Disk-to-puneh transfer of card punch data stored on the disk,
3. Disk-to-tape transfer of system output data overflowing from the disk to magnetic tape,
4. Display of system information.

Off-line operations performed by Pool PPs are utility operations which are initiated by the operator through use of control cards. These operations involve only the peripheral processor and peripheral equipment performing the operation; central memory and the disk are not involved. The off-line operations are as follows:

1. Card to punch,
2. Card to print,
3. Card to tape,
4. Tape comparison,
5. Tape to card,

6. Tape to print,

7. Tape to tape.

Section 5 has additional information on these operations.

Internal operations are initiated by system macros in user programs and may involve central memory and the disk or just the peripheral processor and peripheral equipment performing the operation. The following are typical of these operations:

1. Construction of printer data records in memory for disk transfer (ASPER programs only),
2. Construction of card punch data in memory for disk transfer (ASPER programs only),
3. Memory-to-disk transfers and disk information cataloging,
4. Tape manipulations,
5. Console displays and keyboard input to memory,
6. Execution of ASPER programs.

Peripheral Processor Communication

Figure 4 summarizes the communication flow from one peripheral processor to another. Along the left side of the figure, the arrows show the flow of job assignments. Along the right side, the arrows show the flow of status reports back to the controlling processor.

1.2.4 OTHER EQUIPMENT

Magnetic tape is used by SIPROS 66 for library editing and to store printer and card punch data overflowing from the disk. SIPROS 66 stores all printer and card punch data on the disk until the job has been terminated and then transfers the data to the appropriate output device. If the disk output buffer becomes full during the processing of the job, the overflow print and punch data are written on magnetic tape. As soon as there is disk space available again, the data on the tape are written back on the disk.

SIPROS 66 allows printer and card punch data to be written directly on magnetic tape when large volumes are expected. This is accomplished by specifying an output limit as an installation parameter. Then, if the estimated output exceeds the limit, it is written directly on magnetic tape instead of the disk. Data written on magnetic tape in this way are not written back on the disk when the job is terminated.

The system console unit is used by SIPROS 66 for operator-computer communication. During normal operation, SIPROS 66 displays the status of the jobs in the peripheral processors and in central memory and the usage of input/output channels. Messages are also displayed to the operator when his actions are required, or when requested by executing programs. Inputs are accepted from the keyboard for operator intervention and in response to requests.

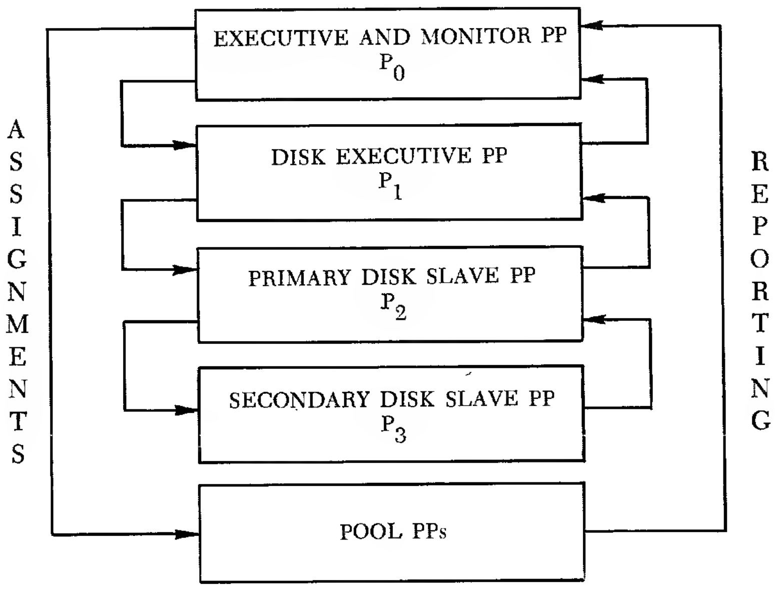

ASSIGNMENTS
EXECUTIVE AND MONITOR PP P_0
DISK EXECUTIVE PP P_1
PRIMARY DISK SLAVE PP P_2
SECONDARY DISK SLAVE PP P_3
POOL PPs
REPORTING

Figure 4. PERIPHERAL PROCESSOR COMMUNICATION

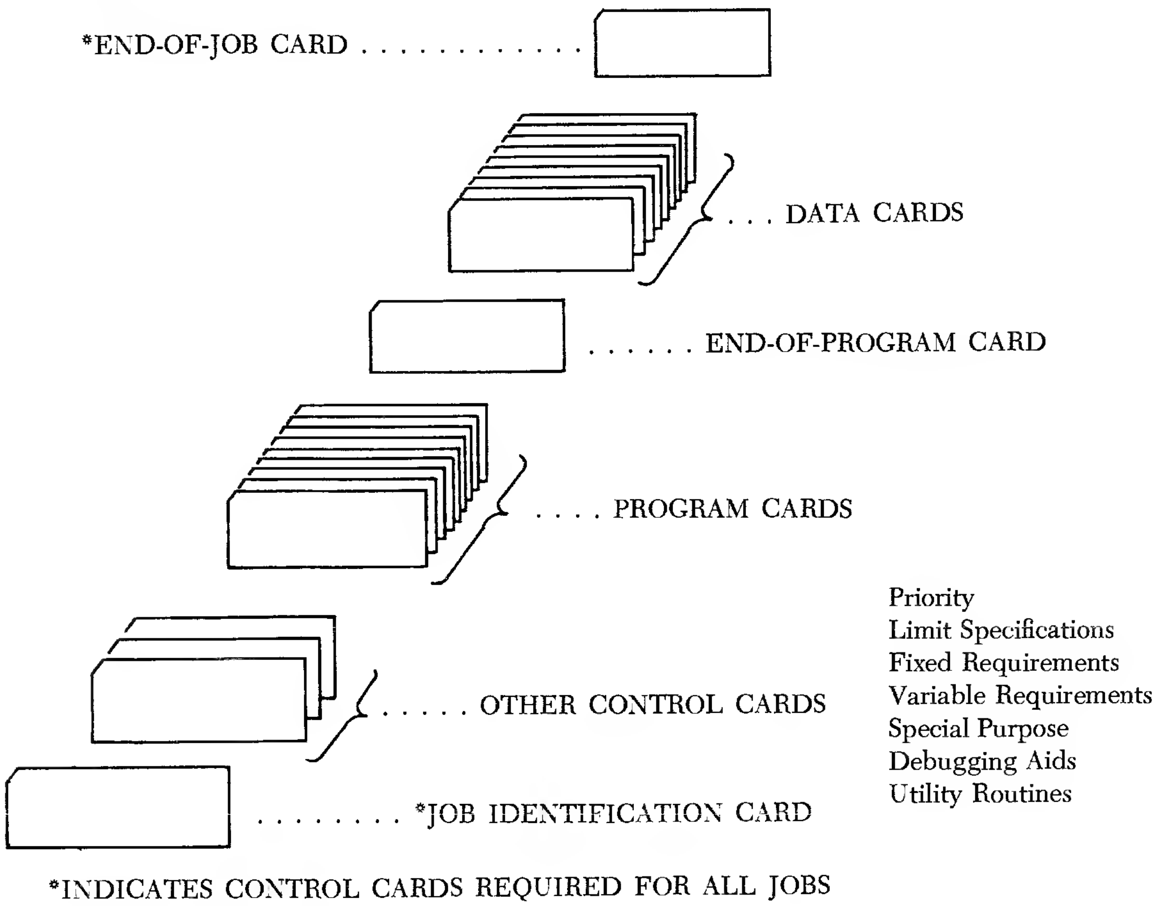
2. JOB ORGANIZATION AND FLOW

The job deck consists of a set of control cards, a deck of program cards, and the input data cards required by the job. Figure 5 shows the composition and organization of a SIPROS 66 job deck.

Every job deck must have at least two control cards: the job identification card and the end-of-job card. The job identification card designates the beginning of the job and is always the first card in the job deck. The end-of-job card designates the end of the job; it will always be the last card in the deck. The remainder of the control cards are used only if called for by the program (e.g. a control card for scratch tape is required only if the program uses scratch tape). These cards are always placed in the job deck between the job identification card and the program cards. The only exception to this is the end-of-program card which is placed between the program and data cards.

Program cards contained in the job deck include source language (FORTRAN, ASCENT, and ASPER) cards for routines to be compiled or assembled and/or binary cards for routines which have already been compiled or assembled. Source language routines may be arranged in any sequence with one exception: ASPER routines must follow the central processor routine with which they are associated. Any source language routine for the central processor may contain FORTRAN and ASCENT program cards intermixed on a card-for-card basis.

Input data cards contain the data to be operated on by the program. They must always be placed after the end-of-program card but before the end-of-job card.



*END-OF-JOB CARD . . . .
. . . DATA CARDS
. . . . . . END-OF-PROGRAM CARD
. . . . PROGRAM CARDS
. . . . . OTHER CONTROL CARDS
Priority
Limit Specifications
Fixed Requirements
Variable Requirements
Special Purpose
Debugging Aids
Utility Routines
. . . . *JOB IDENTIFICATION CARD
*INDICATES CONTROL CARDS REQUIRED FOR ALL JOBS

Figure 5. JOB DECK ORGANIZATION

2.1 JOB INPUT SEQUENCING

Job decks presented to SIPROS 66 are processed in either one of two ways:

1. They are placed on-line in the card reader for transfer directly into the disk job stack; or

2. They are placed in the card reader for off-line transfer by a peripheral processor to magnetic tape. (This tape is formatted in card image and is written in binary mode — odd parity.)

For purposes of discussion, the Control Data® 6600 is assumed to be processing jobs from a full disk job stack or in a wait-for-input condition at the time a set of jobs is made available. When the jobs are ready to enter the operating system, the operator keys in the input unit specification, if necessary, and initiates a ready signal from the card reader or magnetic tape unit. The input specification (card reader or magnetic tape) need not be entered into the system if the input unit is the same as the one previously used. As soon as space in the disk job stack and a peripheral processor are available, the system starts accepting the jobs and loads as many as the disk job stack will hold. The rest of the jobs are accepted without further intervention from the operator as space becomes available. Once a job reaches the job stack, its future processing is determined by the system control cards.

If an off-line card-to-tape transfer is specified, the operation takes place independent of the operating system once the peripheral processor and equipment have been assigned to the operation. Jobs placed on the tape in this off-line operation may be subsequently entered into the system in the manner described in the preceding paragraph.

2.2 CONTROL CARD SPECIFICATIONS

System control cards are used to supply SIPROS 66 with various types of control information. Some of this information, such as job identification or equipment parameters, must be supplied by control cards when required by the system. The balance of the information, such as priority, is left to the discretion of the programmer. If this discretionary information is not supplied by control cards, SIPROS 66 will automatically assign standard system values which have previously been established by the installation.

In addition to supplying SIPROS 66 with control information, system control cards are used to call in debugging aids and standard utility routines. The control cards used for this purpose are described in this section following the description of the cards which are used to supply control information.

Control cards have the following format:

COLUMNS	CONTENT
1	Card identification (8-9 punches)
2-10	Blank
11-72	Control information
73-80	Deck identification

Control information entered into columns 11-72 includes a pseudo-op such as JOB, PRI, etc. and parameters such as name or account number. Commas are used to separate items in each entry and must not be omitted. Blank spaces may precede the commas but may not follow them since the first digit following a comma is always taken as the first digit of the next field. A $ symbol is used to terminate an entry.

Any number of entries (from pseudo-op to $) may be punched into columns 11-72 and they may be in any order with one exception: the JOB entry must be the first entry in the first card. An entry cannot be split between cards.

A period is used after the last entry in a card to indicate that there is no more control information in the card. Comments may be entered after the period. (The only items in the entry that may contain a period are the account number and the priority specification.)

In the following description of control eards, an asterisk (*) indicates the entries required for all jobs. A double asterisk (**) indicates entries that are left to the discretion of the programmer. The remaining entries must be made by control card when required.

2.2.1 JOB IDENTIFICATION

*JOB,NAME,ACC#$

where NAME is 8 characters
ACC# is 10 characters

Example: JOB,POLYNOM,136$

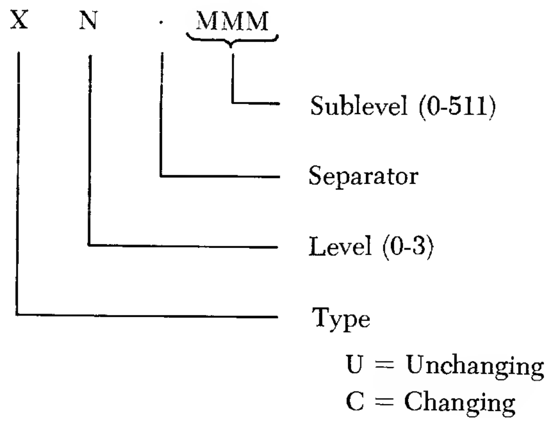
This entry assigns a name and identification number to the job and must not be omitted. The identification number is used by the operator to identify the job once it is in the system.

2.2.2 PRIORITY

**PRI,#1,#2$

where #1 is job run priority
#2 is job I/0 priority

and the specification for #1 and #2 is as follows:



Example: PRI,C2.0,U1.200$

The priority entry is used to assign a priority rating to the job and to the job's I/O operations. If priority is not specified by control card, a system value will be assigned. For additional information, see Section 3.6.

2.2.3 LIMIT SPECIFICATIONS

Central Processor Time Limit

**CPT,MINUTES$

where MINUTES is a non-fraction with a range from 1 through 4095.

Example: CPT,30$

The CPT entry specifies a running time limit for the central processor. If a time limit is not specified by control card, a system value will be assigned.

Peripheral Equipment Time Limit

EQP,DES,TIME,MINUTES,L#$

where DES is the unit designator (ITAPE, PRINT,etc.); MINUTES is the number of minutes (from 1 to 4095) to which the running time of the specified peripheral processor or device is to be limited; and L# is the logical unit number of the processor or device.

Example: EQP,PCP,TIME,15,5$

Peripheral processor logical unit #5 is restricted to a running time of 15 minutes.

This entry specifies the running time limit for the specified peripheral processor or device. If the entry is not made, the unit will not be limited since no system value is assigned (see Section 3.2.2).

Page or Card Limit

**EQP,DES,LIMIT,NUMBER,L#$

where DES is the unit designator (PRINT or PUNCH); NUMBER is the maximum number of pages to be printed or cards to be punched; and L# is the logical unit number of the device.

Example: EQP,PRINT,LIMIT,100,2$

Line printer number 2 is limited to 100 pages for the job.

This entry defines the number of pages or cards to which the designated line printer or card punch should be restricted (see Section 3.2.2). If the entry is not made by control card, SIPROS 66 will assign a system value as a limit.

2.2.4 FIXED REQUIREMENTS

Scratch Tape

EQP,STAPE,L#$

where L# is the logical unit number of a physical tape unit and consists of a decimal number (1 to 3 digits) followed by an A to designate 626 tapes or B to designate 607 tapes.

Example: EQP,STAPE,2A,3A,1B$

This entry defines the scratch tapes to be used by the job and must have an L# for each scratch tape.

Input Tape

EQP,ITAPE,FILE,L#$

where FILE is the reel name or number (6 characters); L# is the logical unit number of a physical tape unit and consists of a decimal number (1 to 3 digits) followed by an A to designate 626 tapes or B to designate 607 tapes.

Example: EQP,ITAPE,432,2A$

The ITAPE entry defines the input tapes to be used by the job and must have an L# for each input tape.

Output Tape

EQP,OTAPE,L#,L#$

where L# is the logical unit number of a physical tape unit and consists of a decimal number (1 to 3 digits) followed by an A to designate 626 tapes or B to designate 607 tapes.

Example: EQP,OTAPE,1A,1B,2B,3B$

The OTAPE entry defines the output tapes to be used by the job and must have an L# for each output tape.

Printer

EQP,PRINT,L#,L#$

where L# is the logical unit number (1 to 3 decimal digits) of one of the printers.

Example: EQP,PRINT,1,4$

The PRINT entry defines the printer to be used by the job and must have an L# for each unit.

Disk

EQP,DISK,L#,L#$

where L# is the logical unit number (1 to 3 decimal digits) of one of the disk units.

Example: EQP,DISK,2,3$

The disk entry defines the disk units to be used by the job and must have one L# for each unit.

Card Reader

EQP,CARD,L#,L#$

where L# is the logical unit number (1 to 3 decimal digits) of one of the card readers.

Example: EQP,CARD,3$

This entry defines the card readers to be used by the job and must have an L# for each unit.

Card Punch

EQP,PUNCH,L#,L#$

where L# is the logical unit number (1 to 3 decimal digits) of one of the card punches.

Example: EQP,PUNCH,2$

The punch entry defines the card punches to be used by the job and must have one L# for each unit.

Peripheral Equipment Exchange

EQP,DES1,L#,EXC,DES2,L#$

where DES1 and L# are the unit designator and logical unit number of the device referenced in the program and DES2 and L# are the unit designator and logical unit number of the device to be substituted for DES1, L#.

Example: EQP,PUNCH,1,EXC,OTAPE,1A$

Wherever the program specifies a punch operation on unit #1, the information will be written on magnetic tape #1.

This entry allows the programmer to substitute one peripheral device for another without requiring a change in the program. The system disk cannot be exchanged for another device (see Section 3.4.1).

Peripheral Processor

EQP,PCP,L#,L#$

where L# is the logical unit number (1 decimal digit) of one of the peripheral processors.

Example: EQP,PCP,5,6$

The PCP entry defines the peripheral processors to be used by user routines and must have an L# for each unit.

Central Memory

**MEM,#$

where # is the estimated number (in 1000s octal) of 60-bit words in central memory required by the job.

Example: MEM,5$

5000 (octal)60-bit words of central memory are required.

This entry allows the programmer to estimate the number of central memory words required by the job. If the estimate is not entered into the system by control cards, SIPROS 66 will attempt to load the job into a standard size block of memory (see Section 3.4.3).

Disk Memory

**DISK,#$

where # is the estimated number (in 1000s octal) of 60-bit words of disk space required by the job.

Example: DISK,2$

2000 (octal) 60-bit words of disk space are required.

The DISK entry enables the programmer to estimate the number of 60-bit words of disk space required by the job. The space estimated by this entry is in the programmer scratch area of the disk. If the entry is not made by control card and there is not sufficient space for the job, the job will be aborted (see Section 3.4.4).

2.2.5 VARIABLE REQUIREMENTS

Peripheral Equipment

EQP,DES,VAR,L#,L#$

where DES is any of the peripheral equipment designators (ITAPE,CARD,etc.) and L# is the logical unit number of a physical unit.

Example: EQP,STAPE,VAR,1A,1B$

This entry is used to define the peripheral devices that may be used by the job in addition to those specified as fixed requirements. It should have L#s for the maximum number of units that might be used. See Section 3.5 for additional information on variable requirements.

Central Memory

**MEM,VAR,#$

where VAR specifies that this is a variable estimate and # is the estimated number (in 1000s octal) of 60-bit words of central memory.

Example: MEM,VAR,10$

10,000 (octal) 60-bit words of central memory may be required by the job in addition to fixed requirements.

This entry allows the programmer to estimate the number of central memory words that may be required by the job in addition to fixed requirements. The exact number of words is determined during the processing of the job (see Section 3.5).

Disk Memory

**DISK,VAR,#$

where VAR specifies that this is a variable estimate and # is the estimated number (in 1000s octal) of 60-bit words of disk space.

Example: DISK,VAR,4$

4,000 (octal) 60-bit words of disk space may be required by the job in addition to fixed requirements.

This entry allows the programmar to estimate the number of 60-bit words of disk space in the programmer scratch area that may be required by the job. This is in addition to fixed requirements. The exact number of words is determined during the processing of the job (see Section 3.5).

2.2.6 END OF PROGRAM AND JOB

End of Program

END$

This card merely identifies the end of the program cards and beginning of data cards. It is not required when there are no data cards.

End of Job

*FINIS$

This card identifies thc end of the job and is required for all jobs. It indicates to SIPROS that all control cards, program cards, and data cards have entered the system.

2.2.7 SPECIAL PURPOSE

Execute Program

EXECUTE$

This entry is used only when a program is to be compiled or assembled and then executed. When all the program cards in the job deck are binary cards, it is not needed.

Compile or Assemble Program

COMPILE$

This entry specifies that there are source language program cards in the job deck to be assembled or compiled. It is not required in a compile-and-execute situation.

Job Accounting

LOAD,ACC$

This control card calls in a special accounting program which assigns charges to the running times and volumes.

LOAD,MTP$

This control card requests SIPROS 66 to print out the job accounting and maintenance test information on the printer.

2.2.8 DEBUGGING AIDS

Error Halt Conditions

The central processor provides for three types of error halt conditions:

1. Address out of range,
2. Operand out of range,
3. Indefinite result.

As an aid to debugging programs, SIPROS 66 allows the programmer to ignore two of these conditions, operand out of range (exponent overflow) and indefinite result, through the use of control cards.

IGNORE,CONDITION$

where CONDITION is

S2 for exponent overflow

S3 for indefinite result

S23 for exponent overflow and indefinite result

Example: IGNORE,S2$

An exponent overflow condition is ignored.

The IGNORE entry causes the specified exponent or indefinite result error halt condition to be ignored. SIPROS 66 continues to process the job when the specified condition occurs.

Memory Dump

DUMP,BA,EA$

where BA is the beginning address and EA is the ending address; both addresses are relative and must be in octal.

Example: DUMP,100,200$

Dump the contents of central memory locations from location 100 through location 200.

The memory dump control card will cause the specified central memory locations to be printed out after a normal or error halt condition. The information is printed out from the disk after the job has stopped.

Memory Map

MAP$

This control card is used to cause a printout of the memory map. The memory map contains the name of each routine in the job, the name of the common area, and the relative address of the location in which each starts. The memory map is printed out at the same time the rest of the printout for the job occurs.

Console Debugging

DEBUG$

The DEBUG control card instructs SIPROS 66 to load the on-line debugging routine into the system. This routine gives the programmer control over the processing of the job and allows him to perform numerous on-line debugging functions. On-line functions which may be performed include:

1. Changing items in the job table such as job status,
2. Entering data into central memory,
3. Changing instruetions in central memory,
4. Dumping all or part of central memory on an on-line printer,
5. Displaying hardware registers and central memory locations,
6. Executing portions of the program.

2.2.9 UTILITY ROUTINES

These control cards call in standard SIPROS 66 utility routines and supply the routines with the required parameters. For a more complete description of these routines, see Section 5.

In the description of each of these control cards, certain symbols are used to define the designators and required parameters for the routines. These symbols are defined as follows:

CTC	Card-to-Punch Routine
CTP	Card-to-Print Routine
CTT	Card-to-Tape Routine
TCM	Tape-Comparison Routine
TTC	Tape-to-Card Routine
TTP	Tape-to-Print Routine
TTT	Tape-to-Tape Routine
CN	is the conversion code and is defined for each control card
FP	is the number of magnetic tape files to be processed (0-511)
FS	is the number of magnetic tape files to be skipped (0-511)
IC	is the input device channel number (in octal)
IL	is the input device record length (0 to 4095)
IS	is the input device synchronizer number (in octal)
IU	is the input device unit number (in octal)
OC	is the output device channel number (in octal)
OL	is the output device record length (0 to 4095)
OS	is the output device synchronizer number (in octal)
OU	is the output device unit number (in octal)
PD	is the type of code following the last record (0 = zeros, 9 = nines)
TM	is the tape mode (0 = binary and 1 = BCD)

Card to Punch

UTILITY,CTC,IC,IS,IU,OC,OS,OU$

Card to Print

UTILITY,CTP,IC,IS,IU,OC,OS,OU,CN$

where conversion codes are:

0 = No conversion

2 = Convert from Hollerith code

Card to Tape

UTILITY,CTT,IC,IS,IU,IL,OC,OS, OU,OL,PD,CN$

where conversion codes are:

0 = No conversion

1 = Convert Hollerith to BCD code

2 = Convert Hollerith to display code

Tape Comparison

UTILITY,TCM,IC,IS,IU,OC,OS,OU$

Tape to Card

UTILITY,TTC,IC,IS,IU,IL,OC,OS,OU,TM, FP,FS,CN$

where conversion codes are:

0 = No eonversion

1 = Convert BCD to Hollerith code

2 = Convert binary to Hollerith code

Tape to Print

UTILITY,TTP,IC,IS,IU,IL,OC,OS,OU, TM,FP,FS,OL,CN$

where conversion codes are:

0 = No conversion

1 = Convert from binary

2 = Convert from display code

Tape to Tape

UTILITY,TTT,IC,IS,IU,IL,OC,OS,OU,TM,FP$

2.3 SAMPLE JOB DECKS

This section illustrates the organization of SIPROS 66 job decks and the use of control cards. Three sample cases are described. They are:

Job Compilation

Job Execution

Job Compilation and Execution

2.3.1 JOB COMPILATION

In this case, it is assumed that the job contains a routine with FORTRAN and ASCENT program cards intermixed on a card-for-card basis, an ASCENT routine, and an ASPER routine which is associated with the ASCENT routine. The only control cards required for the job are the job identification, compile, and end-of-job cards since this is a compile-only operation. Figure 6 illustrates the job deck.

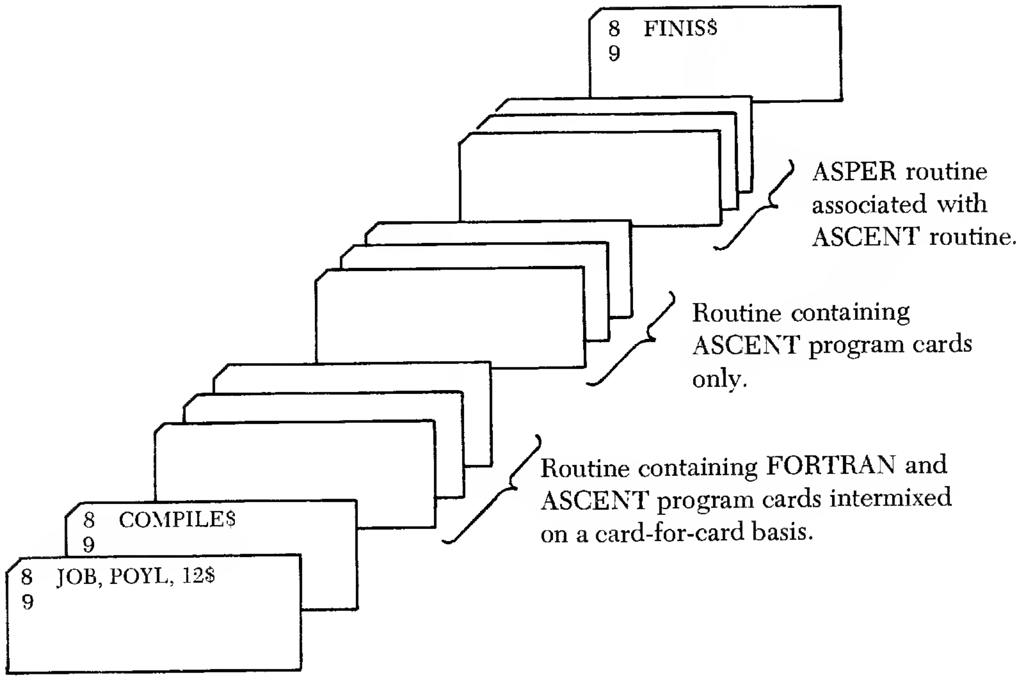

8 FINIS$
9
ASPER routine associated with ASCENT routine.
Routine containing ASCENT program cards only.
Routine containing FORTRAN and ASCENT program cards intermixed on a card-for-card basis.
8 COMPILE$
9
8 JOB, POYL, 12$
9

Figure 6. COMPILE-ONLY JOB

2.3.2 JOB EXECUTION

Here it is assumed that there is only one FORTRAN-ASCENT routine and it is in binary card form ready to be executed. Also, it is assumed that this job uses one input tape and one scratch tape but no line printer other than the system printer. Priority and a central memory estimate are to be specified by a control card. Figure 7 illustrates the job deck.

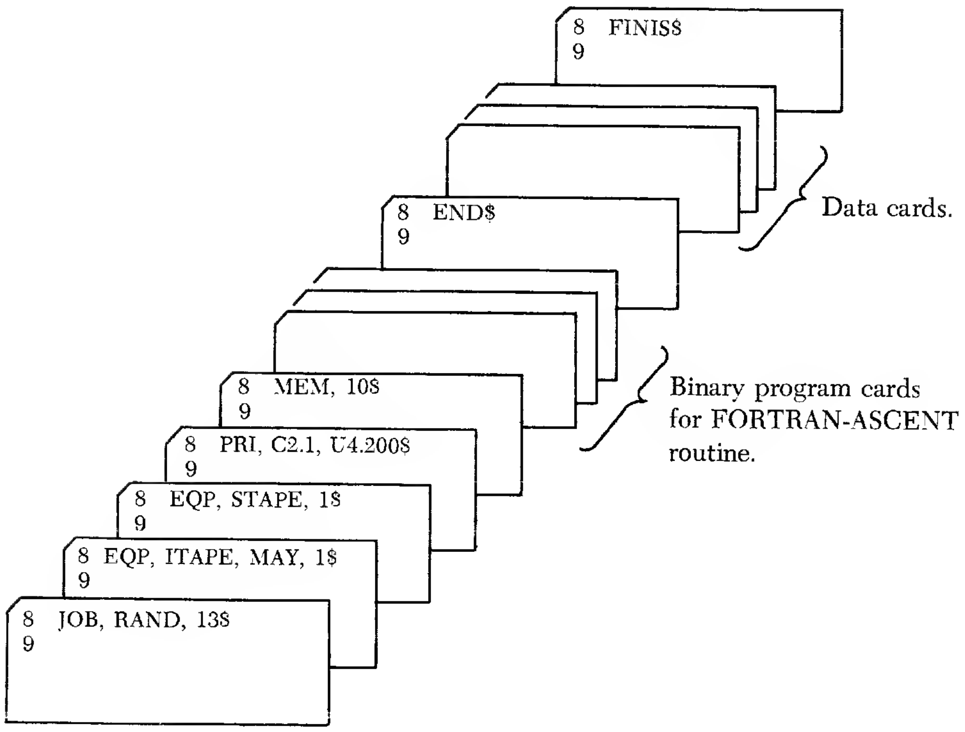
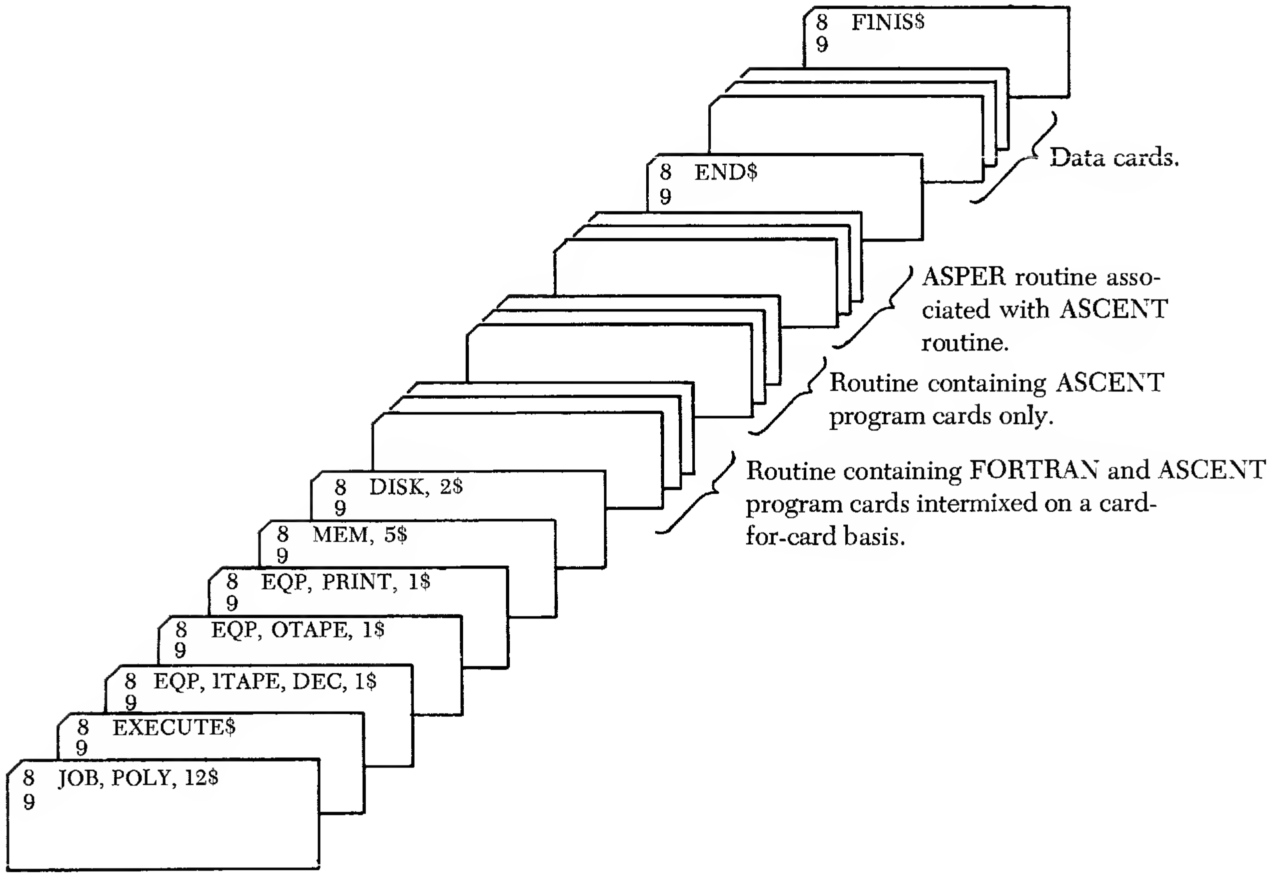


Figure 7. EXECUTE-ONLY JOB

2.3.3 JOB COMPILATION AND EXECUTION

This sample job deck is assumed to be the same as in Section 2.3.1 except that it is to be compiled and executed. In addition, it is assumed that the job requires one input tape, one output tape, and a line printer which is required by the ASPER routine. A priority is to be assigned to the job and to I/O operations, and central and disk memory are to be estimated. Figure 8 illustrates the job deck.



8 FINIS$ 9
Data cards.
8 END$ 9
ASPER routine associated with ASCENT routine.
Routine containing ASCENT program cards only.
Routine containing FORTRAN and ASCENT program cards intermixed on a card-for-card basis.
8 DISK, 2$ 9
8 MEM, 5$ 9
8 EQP, PRINT, 1$ 9
8 EQP, OTAPE, 1$ 9
8 EQP, ITAPE, DEC, 1$ 9
8 EXECUTE$ 9
8 JOB, POLY, 12$ 9

Figure 8. COMPILE-AND-EXECUTE JOB

2.4 JOB FLOW

The SIPROS 66 Batch Loader reads jobs into the disk job staek either from cards or from magnetic tape (see Figure 9). While it is doing this, the Batch Loader also places information supplied by control cards into a job table. This table is continuously examined by the Executive for jobs that are ready to be loaded into central memory.

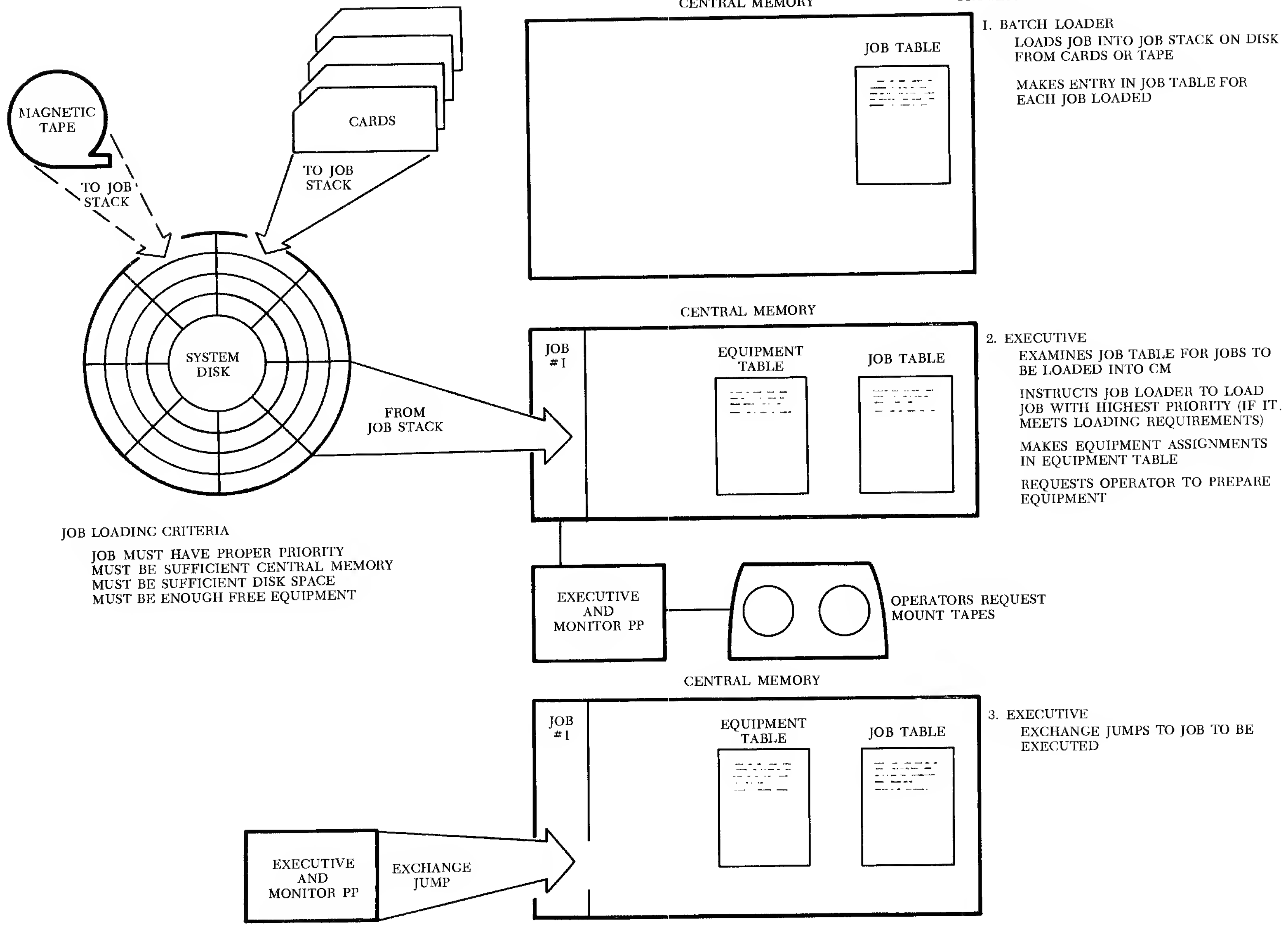
Based upon the control card information in the job table, the Executive picks the jobs with the highest priorities that best fit the available equipment and memory. If a compilation or assembly is required, the first evaluation is based upon the equipment and memory requirements of the programming system. The requirements for execution are considered only after the object code from the compilation reaches the job stack.

Once a job has been selected for execution, the Executive loads the Job Loader into a Pool PP and instructs it to transfer the selected jobs from the disk job stack to central memory. The Executive then makes all equipment assignments for the job. If the disk is called for, disk requirements are coordinated with other needs for its use. If a magnetic tape unit is called for, the Executive puts out a request on the console for the operator to mount a particular tape if it has been specified by a file number on a control card. Then, after the job is in central memory, the Job Loader indicates that the job has been successfully loaded and sets the status of the job to "waiting in central memory." The newly loaded job is given control of the central processor if it has a higher priority than other jobs currently in central memory or if the jobs in central memory that have a higher priority are waiting on I/O requests.

During execution, the program may make input/output requests of SIPROS 66 through the execution of macro instructions. These requests are interpreted by the Central Processor Resident and are presented to the Monitor. The Monitor assigns a Pool PP to the task of providing the service requested. When the resident in the Pool PP receives the request, it loads the required input/output program into the peripheral processor's memory and transfers control to that program. If the central processor program must wait upon completion of the input/output request before it can continue, the Executive removes control from the executing program and gives it to one of the other jobs in central memory. Normally, this job will be the one with the next highest priority providing it does not have I/O waits.

Printer and card punch output from a central processor program is placed in a central memory buffer by the CP Resident and is transferred to the disk output buffers. When the job has been completed and the printer or card punch is free, the Executive assigns the first available peripheral processor – normally the one that has just completed the last print or punch job – to the task of printing or punching the results from the disk.

If a job requires compiling or assembling, the 6600 Programming System is loaded. During this phase, the source deck is drawn from the job stack as data for the programming system. The same procedures are used by this system as are used by other programs. Outputs from the 6600 Programming System may be source listings, assembly listings, and/or the binary programs. The binary program can be routed to the job stack for scheduling and execution and/or to be punched on binary cards. Any data routed to the printer or card punch are retained on the disk (or tape, in overflow cases) to be combined with program results during execution.



PROCESSING STEPS

1. BATCH LOADER
   LOADS JOB INTO JOB STACK ON DISK FROM CARDS OR TAPE

   MAKES ENTRY IN JOB TABLE FOR EACH JOB LOADED

2. EXECUTIVE
   EXAMINES JOB TABLE FOR JOBS TO BE LOADED INTO CM

   INSTRUCTS JOB LOADER TO LOAD JOB WITH HIGHEST PRIORITY (IF IT MEETS LOADING REQUIREMENTS)

   MAKES EQUIPMENT ASSIGNMENTS IN EQUIPMENT TABLE

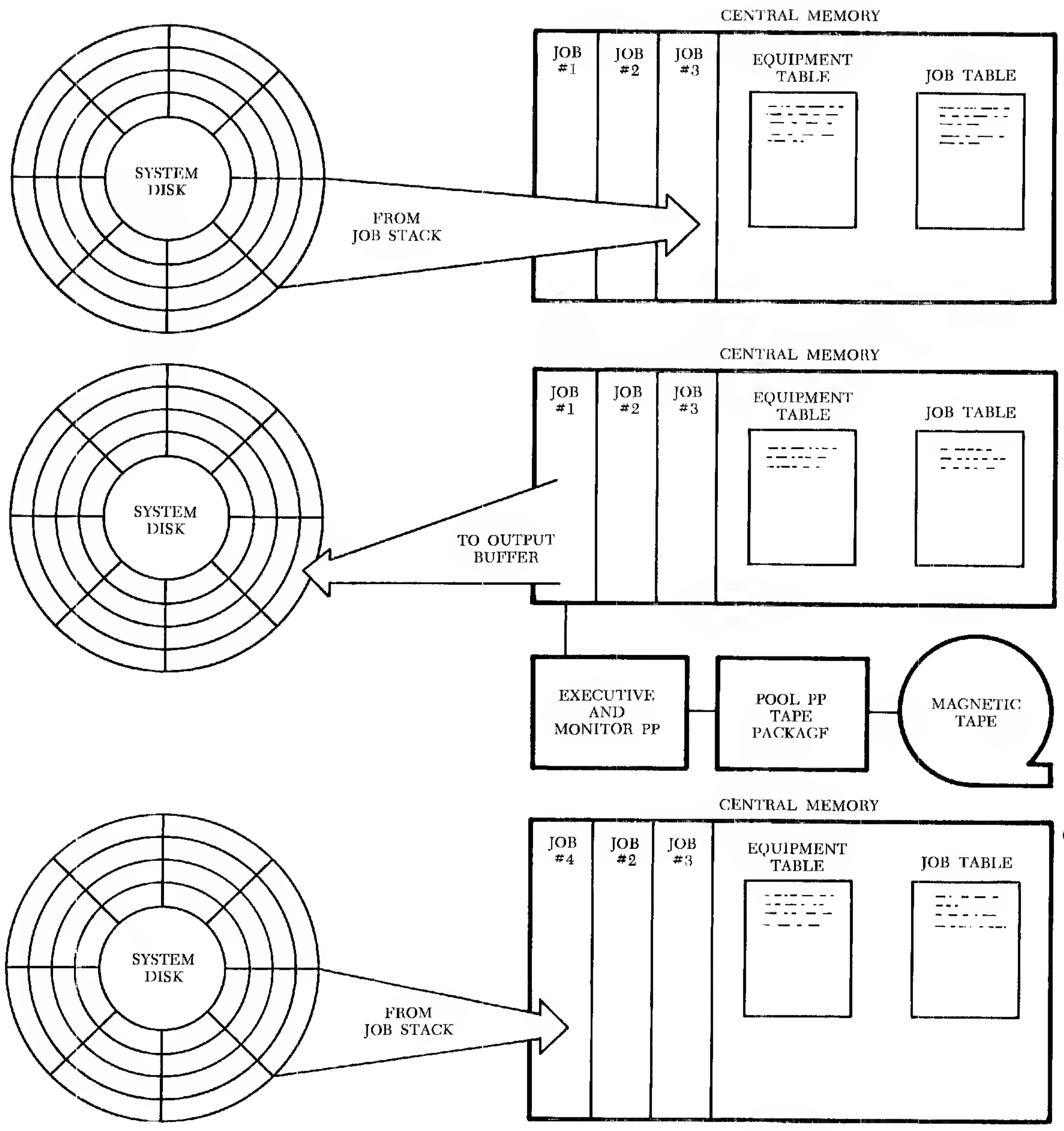
   REQUESTS OPERATOR TO PREPARE EQUIPMENT

3. EXECUTIVE
   EXCHANGE JUMPS TO JOB TO BE EXECUTED

JOB LOADING CRITERIA

JOB MUST HAVE PROPER PRIORITY
MUST BE SUFFICIENT CENTRAL MEMORY
MUST BE SUFFICIENT DISK SPACE
MUST BE ENOUGH FREE EQUIPMENT

Figure 9. JOB FLOW, PART 1



4. EXECUTIVE

INSTRUCTS JOB LOADER TO LOAD OTHER JOBS INTO CENTRAL MEMORY UNTIL IT IS FULL

MULTIPROCESSES JOBS IN CENTRAL MEMORY

5. EXECUTIVE

DIRECTS OUTPUT DATA FOR PRINTER AND PUNCH TO OUTPUT BUFFER ON DISK

DIRECTS OUTPUT DATA FOR TAPE TO POOL PP WHICH WRITES TAPE

6. EXECUTIVE

SCHEDULES NEW JOB FOR CM WHEN JOB TERMINATES

INSTRUCTS JOB LOADER TO LOAD NEW JOB FROM JOB STACK ON DISK INTO CM

Figure 9. JOB FLOW, PART 2

3. JOB PROCESSING

When processing jobs, SIPROS 66 follows certain prescribed methods in identifying jobs, in limiting jobs to a given time or output, in allocating memory and equipment, in assigning a processing sequence to jobs, and so on. These processing methods are described in this section.

3.1 JOB IDENTIFICATION

Jobs being processed by SIPROS 66 are doubly identified, externally and internally. Every job has an external job identification which is supplied by the job identification card on the front of each job. When reference is made to a job by an external source such as the display console, it is this external job number which is used. The internal identification is a number assigned to a job by SIPROS 66 at the time it is entered into the system. This number is not available to programmers since it is used only to identify the job within the various parts of SIPROS 66.

3.2 LIMIT SPECIFICATIONS

SIPROS 66 allows separate running time limits to be assigned to the central processor program and to input/output programs. It also permits the assignment of a fixed volume of printed or punehed output for the job.

3.2.1 CENTRAL PROCESSOR TIME LIMIT

A central processor time limit may be specified by a system control card. This time limit is the amount of time needed by the central processor to complete this job, not taking into account input or output time. If no specification is made by control card, the time limit will be set at a standard value. This standard value can be chosen to fit the requirements of the installation and may, at one extreme, be a small value to prevent undue waste from program error loops, or, at the other extreme, be a large value that effectively removes the limitation. If the limit is exceeded, the program is dumped and the appropriate error indication is given on the job log.

3.2.2 INPUT OR OUTPUT LIMIT

An input/output time limit may also be specified by a system control card to limit the time the job spends in an I/O loop (except for a print or punch loop). For each peripheral processor or device to be limited, a logical unit number and time limit are specified. Then, if the limit is exceeded, the program is dumped and the appropriate error indication is given on the job log. If a limit is not entered into the system by a control card, the I/O operation will not be limited since SIPROS 66 does not assign standard system values as I/O time limits.

In addition to time limits on other I/O equipment, card punch and line printer equipment may have output limits imposed on them through the use of control cards. These limits specify the number of cards to be punched or pages to be printed. Standard system values will be assigned if the limits are not specified by control cards. Whenever the card punch or line printer output exceeds the limit, the program is dumped and the error indication is given on the job log.

3.3 MEMORY DUMP

All or part of the central memory area of the job may be dumped after a normal or error halt condition occurs. This is accomplished by inserting the memory dump control card, which specifies the area to be dumped, into the job deck. The content of the specified area is then printed out from the disk after the job has been terminated.

3.4 FIXED EQUIPMENT REQUIREMENTS

For most jobs, it is possible to specify the exact tape, memory, and disk requirements before the jobs are processed. Equipment is assigned to these jobs at the time they are loaded into central memory.

3.4.1 PERIPHERAL EQUIPMENT ASSIGNMENT

With the exception of the disk, the peripheral equipment necessary for the execution of a job is specified on the equipment request control cards (those with an EQP pseudo-op). The logical unit numbers used to reference the devices are assigned to physical units by SIPROS 66 at the time the job is loaded into central memory. All of the equipment specified for the job must be available before SIPROS 66 allows the job to be loaded.

SIPROS 66 allows peripheral equipment to be exchanged at load time by specifying the units to be exchanged on an exchange control card. Any two units other than the system disk may be exchanged as long as they are compatible (e.g. a line printer cannot be exchanged for a card reader). The following list illustrates compatible exchanges:

Card reader for tape

Tape for card reader

Printer for tape

Tape for printer

Punch for tape

Tape for punch

3.4.2 PERIPHERAL PROCESSOR ASSIGNMENT

The allocation of peripheral processors (other than system processors) to a job is accomplished by control cards in much the same manner as is the allocation of other peripheral devices. However, this allocation does not include the actual assignment of the peripheral processor to the job but merely precludes the loading of combinations of jobs that could need more than are available. Actual assignment is made dynamically during the processing of the job by system macros which cause the processors to be loaded and reloaded with specialized peripheral processor programs. Five peripheral processors can be assigned in this manner to specific user jobs at the same time.

One of the features of this method of assignment is that it allows peripheral processors to be assigned to special user programs which have been written in the ASPER language. Again, the assignment is made dynamically during the processing of the job by a system macro (see Section 4.6). After the assignment has been made and the special user program is in operation, SIPROS 66 can be used to process other jobs in the normal manner providing the special program is not using the full potential of the 6600.

As an example of the assignment of peripheral processors to special user programs, assume the following: In addition to the jobs in central memory which use the system I/O programs, there are three special peripheral processor programs. One is a program in which no input or output is involved (Case 1 in Figure 10). The peripheral processor is merely performing an auxiliary function such as editing for a central processor program. The second special program (Case 2 in Figure 10) is a real-time program which is controlling and transferring real-time information for a central processor program. The last special program (Case 3 in Figure 10) is transferring information from normal jobs into and out of the system over remote devices.

3.4.3 MEMORY ALLOCATION

An estimate of the memory required to execute a job can be given as a control card parameter. This estimate is used by SIPROS 66 in scheduling the job for loading. If no estimate is given then SIPROS 66 attempts to load the program in a standard size block of memory. The size of this block is established by the installation. If the program is larger than the standard size block, the loading process is pseudo-completed and the total memory needed is recorded for rescheduling of the job.

When the system is in operation, a variety of jobs can be making demands on central memory. As the central memory load fluctuates (by some jobs terminating or by new jobs entering the system), the system automatically reallocates memory, if necessary, to provide space for new jobs. The reallocation is necessary whenever the following two conditions exist: no one block of unused memory is large enough for any job in the disk job stack; and, there are unused blocks of memory which could, if placed together, provide a block which is large enough to load a job. Even though reallocation is necessary, it will not take place if the central memory program that has to be relocated has a special peripheral processor program in operation. Standard input-output programs initiated by system macros, however, will not prevent reallocation of memory when they are in operation.

If a central memory program requires memory expansion during execution, the variable amount of memory must be specified by control card (see Section 3.5). SIPROS 66 then allows the program to request additional memory up to the amount specified.

A central memory program may release memory that was previously estimated as a fixed requirement. The space released is always at the high end of the job area and is in 512-word blocks. A system macro is used for this purpose (see Section 4.6).

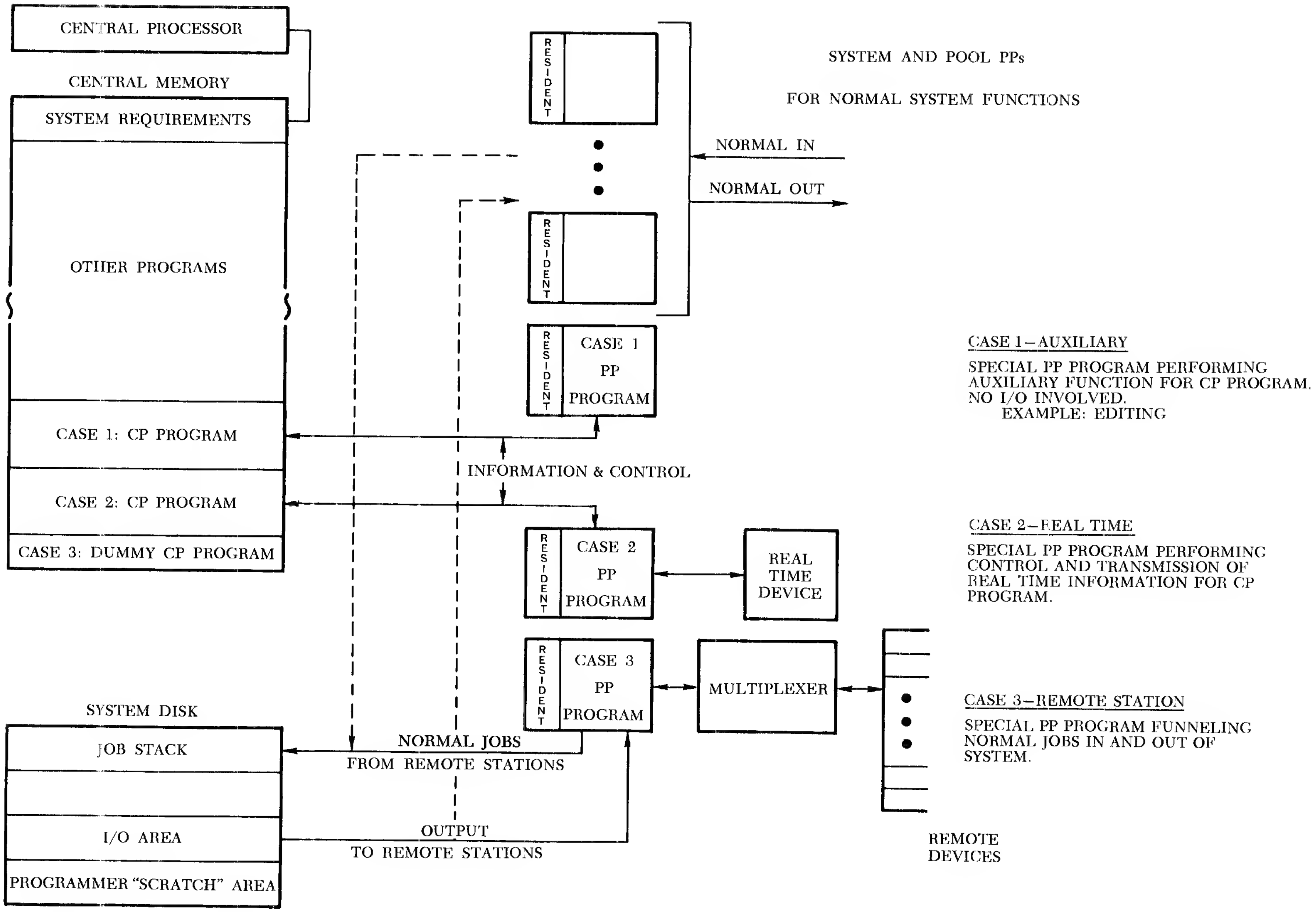


Figure 10. SPECIAL PERIPHERAL PROCESSOR PROGRAMS

3.4.4 DISK ALLOCATION

SIPROS 66 provides space assignment, information access, and accounting services for jobs which use the disk units. Since there very likely can be several jobs in the job stack and in central memory needing disk space, a means must exist to determine the disk space allocation required by each job to avoid saturation of the disk unit's capacity in the midst of an execution phase. There are two methods available to determine the disk requirements of a job. One is simply not to mention anything about it. SIPROS 66 attempts to load the job. The disk space requirements for each routine within the job are determined by the compiler or assembler. During the loading process, the disk requirements for each job are determined. If there is sufficient space available on the specified units, the loading process proceeds normally. If there is not, the loading operation is aborted and the job is rescheduled for loading at a time when there is sufficient space. A second provision allows for an estimate of disk requirements by means of a control card. Exercise of this option lessens the chance of the load abort. On all but the first run, the exact amount of disk space used by each job, which was obtained during the previous run, is available in the job table.

3.5 VARIABLE EQUIPMENT REQUIREMENTS

There are certain classes of jobs for which it is difficult to specify the exact tape, memory, and/or disk requirements. One class, which probably represents the exception, consists of recursive procedures. Equipment requirements arising during the processing of recursive procedures are dependent upon the class of data encountered. For this reason, exact requirements are not known until the job is in execution. Exceptional situations such as these necessitate specifying a fixed requirement (which would satisfy the majority of cases) and also a variable requirement which is determined during processing.

SIPROS 66 provides a means of requesting additional memory, disk storage, or magnetic tapes while jobs are in process. These requests require special control because it is possible for several jobs of this class to be in process concurrently. This could bring about a conflicting situation in that each job in central memory could not continue until additional units were assigned. SIPROS 66 provides this control. All the programmer has to do is to specify fixed and variable (maximum) amounts of memory, disk storage, and magnetic tape. Then, during execution when a program is ready to use equipment defined as variable, a request for that equipment from the system is made by the appropriate system macro (section 4.6). SIPROS 66 maintains the status of such requests to prevent an equipment deadlock from occurring.

SIPROS 66 also provides a means of releasing memory that has been allocated to the job as a variable requirement. Again, this is done by using a system macro (see Section 4.6) in the central processor program. The memory released will be from the high end of the job area and will be in 512-word blocks.

3.6 PRIORITY

The priority of a job determines the order in which SIPROS 66 processes that job and allocates equipment to it. In SIPROS 66, it is possible to assign one priority to the job and a separate priority to I/O operations. These priorities are normally specified by the operator through the use of control cards but will be set up by the system if control cards are not used. When the system assigns the priorities, it uses a value previously defined as an installation parameter for job priority then equates I/O priority to that value.

The SIPROS 66 system provides for two types of priority: changing and unchanging. A changing priority is periodically increased as long as the job is waiting in central memory or in the job stack. This automatic upgrading of the priority rating ensures that a job with a relatively low priority it not delayed indefinitely by programs with higher priorities. Eventually, the job has the highest rating and is then processed. Note that this automatic upgrading of priority results in alternate processing of jobs. As an example, assume that there are two jobs in the system with the same priority and that one job is waiting while the other is running. Only the priority of the job which is waiting will be upgraded. As a result, it will take control away from the job being processed which will, in turn, become the waiting job and will have its priority increased. This alternate processing will continue as long as both jobs are in the system. An unchanging priority differs from a changing priority in that it allows a job to remain at a fixed rating for an indefinite period of time. Both types of priority may be assigned to the job and to the I/O operations for the job.

For either of the two types of priority, there are four priority levels. Within each level, there is a range of values from 0 through 511. Figure 11 illustrates a 12-bit priority word.

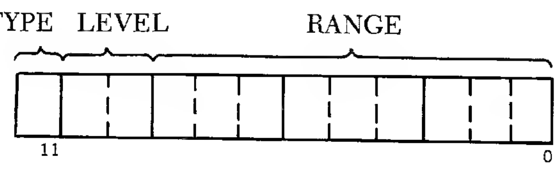


Figure 11. 12-BIT PRIORITY WORD

A 1 in bit position 11 indicates an unchanging priority; a 0 indicates a changing priority.

In the SIPROS 66 system, an installation may specify whether a changing priority is to be upgraded across levels or is to stay within its own level. Also, it may set an arbitrary limit in the fourth level above which no changing priority may go. This makes it possible for the programmer to specify a high (unchanging) priority job which will maintain control of the system until it has been completed. It should be noted, however, that only job priority may be restricted to specific levels. I/O priority will always increase across levels to the limit set in the fourth level.

Figure 12 summarizes the structure of priorities used in the SIPROS system.

3.7 CRITERIA FOR LOADING A JOB

In order for a job to be selected and loaded into central memory for execution, several criteria must be satisfied. They are:

1. The job must have the highest priority.
2. There must be sufficient contiguous central memory space available to meet fixed and variable requirements. Either the standard requirement or an amount equal to that estimated for the job on its control cards must be met.
3. There must be at least as much free space on the disk as is estimated as fixed and variable requirements on the job's control cards.
4. There must be enough free equipment of the type specified to meet estimated fixed and variable requirements.

All of these criteria are determined from information obtained from control cards accompanying the job or they are assumed to be standard.

3.7.1 COMPILATION OR ASSEMBLY ONLY

For jobs containing only routines to be compiled or assembled, the criteria for job loading, with the exception of priority, is not specified explicitly on the control cards. This is because these criteria are the same for all jobs of this type and are supplied from the programming system to SIPROS 66. When all the criteria are met, the compiler-assembler is transferred into the central memory to process the job. From this point on, the job is treated as input data to the compiler-assembler program.

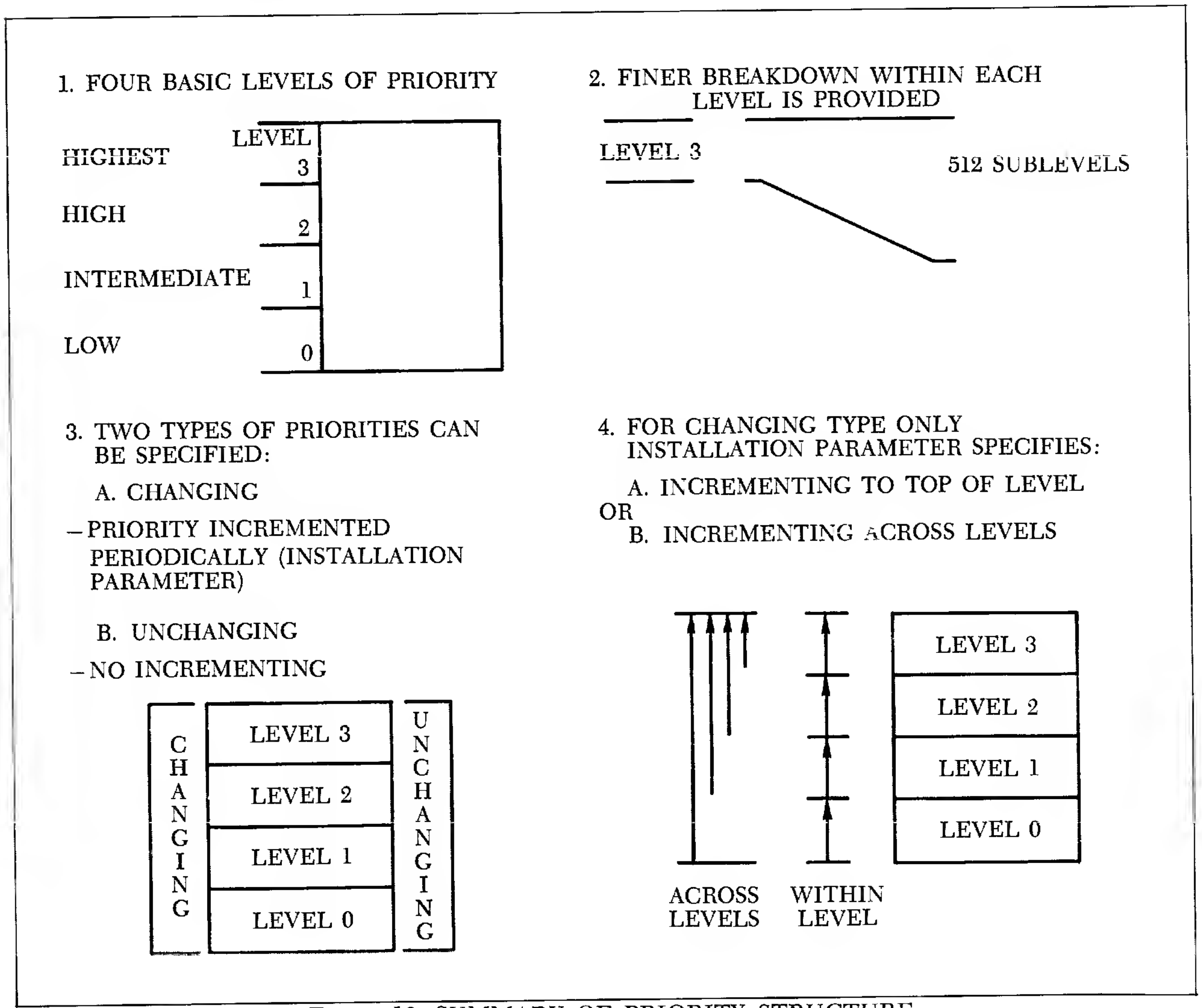
3.7.2 EXECUTION ONLY

For jobs which are already compiled, the job loading criteria can be explicitly specified on the job's control cards. When all the criteria are met, the job is transferred by means of the Job Loader into the central memory space allocated to it by SIPROS 66. Jobs of this type are assumed to be made up of independently compiled or assembled routines in relocatable binary card deck form. These routines are read into central memory and relocated according to the control information included with each routine and according to special punches on each card in each deck (see Appendix 5).

Each routine is loaded into memory consecutively, i.e., each is loaded into the location immediately following the last location used by the routine immediately preceding it. Use is made of the subroutine library to obtain additional subroutines needed by the job but not contained in the job deck. As each routine is being loaded, COMMON and disk storage area references of the same name are linked together. Once the job has been loaded successfully, its routines are linked together through the calling sequences by the Job Loader. This completes the loading procedure for a normal unsegmented job.

In cases where a job is very large, too large to be conveniently loaded into memory at one time, provision has been made to break up the job into pieces or segments. The segments are of two types: basic and normal. There can be only one basic segment for a job; it contains a set of routines and storage areas that are never affected by segment overlay operations. Normal segments contain the routines and storage areas that are transferred to core as a result of overlay requests. There can be any number of normal segments in a job. Each normal segment, upon request, is loaded by the Job Loader into memory immediately following the basic segment. Listed below are some of the more important characteristics of the handling of segmented jobs:

1. The request for an overlay of a normal segment can occur in either type of segment present in memory.
2. An overlay request initiates a loading process which is similar to that used for an unsegmented job, thereby assuring compact memory usage with only one copy of each routine in memory.
3. Segments are defined at execute time by special control cards containing the segment identifiers and a list of the routines or other segments in each of them.
4. Only one copy of each routine needs to be loaded with the job, even though it is used by several segments.
5. Control transfers may be indicated by the overlay request and defined at execute time.


1. FOUR BASIC LEVELS OF PRIORITY
HIGHEST
LEVEL 3
HIGH
2
INTERMEDIATE
1
LOW
0
2. FINER BREAKDOWN WITHIN EACH LEVEL IS PROVIDED
LEVEL 3
512 SUBLEVELS
3. TWO TYPES OF PRIORITIES CAN BE SPECIFIED:
A. CHANGING
– PRIORITY INCREMENTED PERIODICALLY (INSTALLATION PARAMETER)
B. UNCHANGING
– NO INCREMENTING
CHANGING
LEVEL 3
LEVEL 2
LEVEL 1
LEVEL 0
UNCHANGING
4. FOR CHANGING TYPE ONLY INSTALLATION PARAMETER SPECIFIES:
A. INCREMENTING TO TOP OF LEVEL
OR
B. INCREMENTING ACROSS LEVELS
ACROSS LEVELS
WITHIN LEVEL
LEVEL 3
LEVEL 2
LEVEL 1
LEVEL 0
Figure 12. SUMMARY OF PRIORITY STRUCTURE

3.7.3 COMPILE AND GO

Jobs of this type are a combination of the previous two types. These jobs are handled in two phases. The first phase treats the job almost as if it had only routines to be compiled or assembled. When the first phase is complete, the relocatable binary decks for the newly compiled routines will have been combined with those loaded with the job. This combined deck is used for phase two which treats the job as an execution-only job since it no longer contains any routine to be compiled. Thus, jobs of this type are handled by SIPROS 66 almost as if there were two jobs to be run consecutively.

3.8 INPUT DATA

Input data decks are placed in the card reader following the end-of-program card and are loaded onto the disk along with the program deck. When the program first executes a read card macro (see Section 4.4) the system traps the request and loads the appropriate data from the disk into a central memory buffer area reserved by the assembler or compiler for card input data. Subsequent requests for card reading cause data to be transferred from the central memory buffer to the program data region. After the request that removes the last block of data from the central memory buffer is processed, the system loads another block of data from the disk. (The size of the central memory buffer is a parameter which is established by the installation and is the same size as the disk record – from 512 to 4096 60-bit words in increments of 512.)

3.9 JOB ACCOUNTING

The job account number must appear on the identification card that precedes each job deck. Absence of this identification number results in the immediate termination of the job since it is used by SIPROS 66 to associate various equipment usages, etc. with the proper program.

At the beginning and at the end of each event during the processing of a job, accounting information is logged in a job log on the disk. In addition, as maintenance test routines are used, the results of these tests are placed in the job log. This log contains:

- Job Name
- Job Account Number
- The time of assignment and elapsed time for the following:

  Central Processor

  Peripheral Processor

  Peripheral Devices

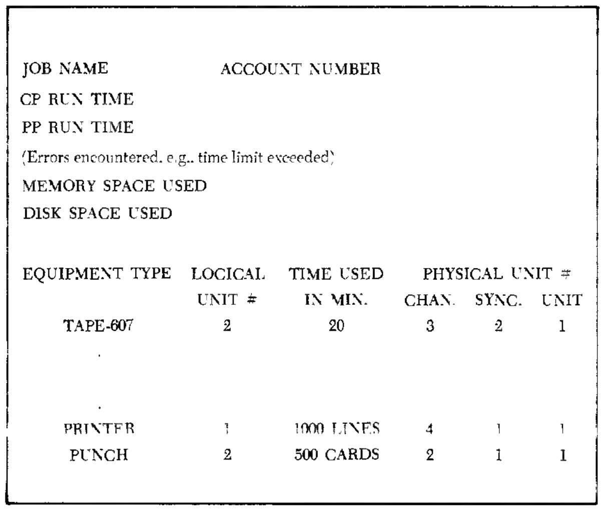
  Memory

  Disk

  The elapsed time for a peripheral device is the total time a peripheral processor is tied up by that device.
- Diagnostic information encountered by the system routines during processing.

The job accounting information for each program is automatically printed out from the disk on the last page of output for the job (see Figure 13). Operating personnel may also make requests through the console keyboard for all or portions of the job accounting log to be displayed (see Figure 14), printed, or written on magnetic tape. In all cases, the information is retained on the disk after output.

Should the job accounting log on the disk become full, both the job accounting information and maintenance test information will be printed out or written on magnetic tape. However, sinee the size of the disk log is determined by the installation, situations such as this will not normally occur.



```
JOB NAME                ACCOUNT NUMBER
CP RUN TIME
PP RUN TIME
(Errors encountered, e.g., time limit exceeded)
MEMORY SPACE USED
DISK SPACE USED

EQUIPMENT TYPE   LOGICAL   TIME USED     PHYSICAL UNIT #
                 UNIT #    IN MIN.     CHAN.  SYNC.  UNIT
    TAPE-607        2         20         3      2      1
       .


       .
    PRINTER         1     1000 LINES     4      1      1
     PUNCH          2     500 CARDS      2      1      1
```

Figure 13. PRINTER FORMAT OF JOB ACCOUNTING INFORMATION

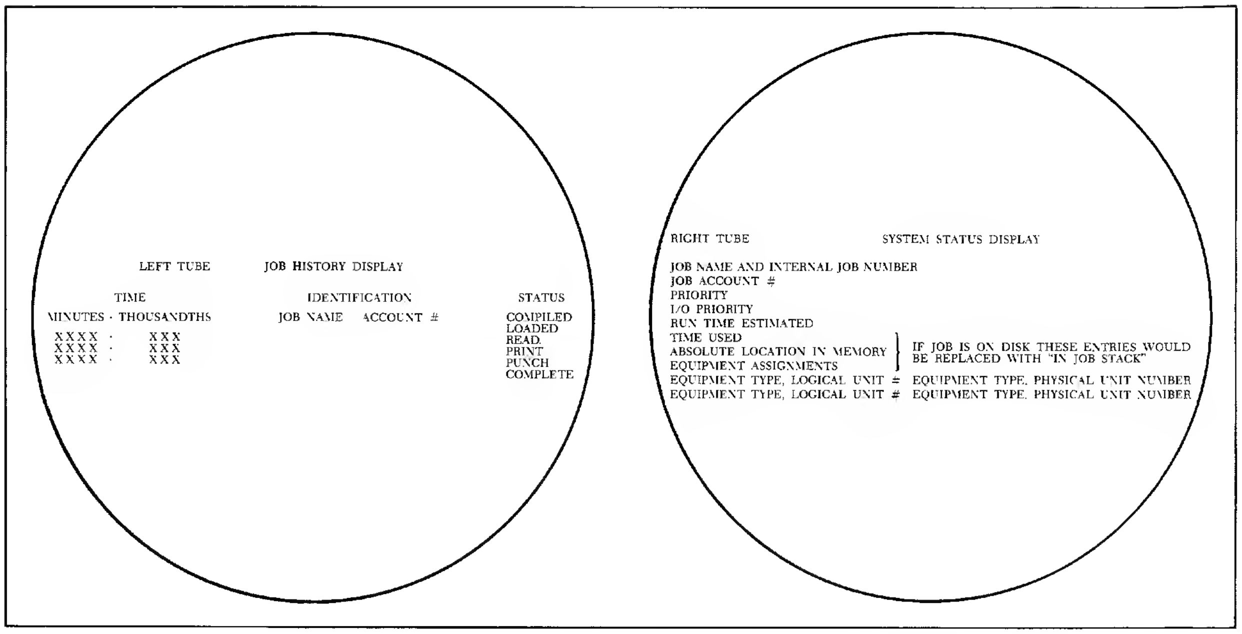


Figure 14. DISPLAY FORMAT OF JOB ACCOUNTING INFORMATION

A special job accounting program may be written at each installation to assign charges to processing times and volumes. This program is called into memory by a control card.

3.10 OUTPUT

The programmer must specify, as a control card parameter, the logical unit number of output tapes which are to be saved. Output to the printer or card punch is stored on the disk until completion of the job. If the output limit for the operation exceeds a system limit as specified at the installation, the print or punch data are sent to magnetic tape. Upon termination of the job, SIPROS schedules the print or punch output to the appropriate peripheral device. A peripheral processor transfers the final output to the peripheral device.

3.11 STOP CONDITIONS

There are two kinds of central processor stop conditions recognized by the Control Data® 6600 and thus by SPIROS 66.

. Normal Stop

. Error Stops

Normal Stop

This is the normal means of terminating a job and returning control to SIPROS 66. This is accomplished merely by executing the STOP instruction.

Error Stop

There are three types of error stops recognized by the 6600. They are:

1. Address out of range — an attempt to reference memory location outside the area allocated to this job.
2. Exponent overflow — The maximum size of the exponent of a floating point number was exceeded during a floating point operation.
3. Indefinite result — A floating point operation has occurred which resulted in an indefinite result. An example of this would be an attempted division by zero.

An option is provided for the programmer to ignore either or both of the last two error stop conditions. In case of an error stop the programmer may, at his option, specify either one or both of the following items:

1. Map (list of names and memory locations of subroutines) of the job's allocated memory area.
2. Dump of the job's allocated memory area.

3.12 INSTALLATION PARAMETERS

Wherever possible, SIPROS 66 has been designed to give the installation maximum flexibility in selecting the various system values which define central and disk memory allocation and processing limits. These values are selected by the installation in one of two ways: by entering the values as control card parameters or by entering them as installation parameters. Control card parameters were described in Section 2.2. Installation parameters are described in this section.

Installation parameters are entered as card input into the system at the time SIPROS 66 is loaded into the 6600 and may be subsequently altered only by reloading. The following lists specify the parameters that may be selected by each installation.

CENTRAL MEMORY PARAMETERS

1. Memory size 131,072 or 65,536 60-bit words.
2. Total system requirements.
3. I/O buffer size (512 60-bit word minimum).
4. Space allotted to system routines.
5. Space allotted to trial load of job if memory estimate not specified by control card.
6. Size of job table area.

DISK MEMORY PARAMETERS

1. Space allotted to library functions.
2. Space allotted to output area.
3. Space allotted to job stack.
4. Space allotted to programmer scratch area.
5. Physical (or fixed) record size (from 512 to 4096 60-bit words in increments of 512 60-bit words).

PROGRAMMING SYSTEMS PARAMETERS

1. Symbol table size.
2. Temporary storage region.
3. Programmer macro storage.

OPERATION PARAMETERS

1. Standard installation priority.
2. Central processor execution time limit.
3. Print or punch output limit.
4. Programmer display time limit.
5. System balance parameters for peripheral processors.

4. SYSTEM MACROS

System macro instructions provide communication links between an ASPER or central processor program and system peripheral processors. While most of these macros direct the operating system to perform input/output operations, others request equipment assignment, check the status of external operations, produce program overlays, use system peripheral processors in conjunction with an ASPER or central processor programs, and request the operating system to provide channel scheduling services.

The communication link provided by the system macros allows a two-way information transfer. The ASPER or central memory routine not only gives the system peripheral processor request information but also a location in central memory in which the system peripheral processor enters the status of the requested operation, reporting its success back to the ASPER or central processor routine. Each system macro must have a status response word which is set by the operating system in performing the function of the individual macro request.

All communication links are made through central memory. The status response word identified by the request must be a central memory word. Similarly, requests for the system to input or output data for an ASPER or central processor program assume the data region is located in central memory. The regions required may be defined either by the central processor program, if one is used, or by the ASPER BSSCM pseudo operation.

When applicable, system macros provide a buffered and a non-buffered mode. In the buffered mode, where the macro is used without an appended "W," the ASPER or central processor routine is free to continue processing while waiting for the results requested. However, it must do its own status checking, by means of another macro, to determine when the requested operation is completed. In the non-buffered mode, where "W" is appended to the macro opcode, the macro itself determines when the requested operation is completed or aborted, and the ASPER or central processor program stops until the results are known. Both modes return full information on the status of the request.

ASCENT generates a sequence of code from the system macro which initiates the requested system function. The Return Jump generated is followed by the parameters in line with the object code. The parameters may be any of the following forms:

Constant (integer or octal) – specifies the parameter itself, such as a unit number, the record length, or the conversion mode.

Symbolic Location – specifies the location which contains the parameters.

Literal – specifies the parameter itself. ASCENT places in a location at the end of the object code the parameter specified by the literal.

Name – certain macros require a file or program name. The NAME is converted to display code and becomes the parameter.

An example of a macro follows:

```
RDCW 1, (S), (BA), (BA+8), 8, 2
```

Assume: S = 1001

BA = 2500

Then: ASCENT generates a location for each literal specified by the macro. If the end of the object code is location 4200, then:

```
4201 = 0 . . . 01001
4202 = 0 . . . 02500
4203 = 0 . . . 02510
```

The communication link in the object code for the Central Resident program becomes:

	59 30	29 0	
P	RJ SUB		
P+1	EQ B0 B0 P+5	OP N	
P+2	00... 01	40...04201	
P+3	40... 4202	40...04203	Parameters
P+4	00... 0010	00...00002	
P+5	OBJECT CODE		

SUB — A routine that forms the parameters in locations 000002-00000 (N+1) for communication with the operating system. Location 000001 contains the operation code that the macro requested.

OP — Operation code assigned by the system to each macro

N — Number of parameters.

For each system macro encountered, ASPER generates a sequence of coding which communicates the requested function to the system through the peripheral processor resident program. The coding consists of a Return Jump to the resident routine followed by an unconditional jump past a vector of words containing the octal opcode, buffer-mode flag, and evaluated parameters.

A list of required parameters is specified for each system macro. These parameters may be written in various forms depending on the type of parameter. Parameters representing peripheral processor locations which contain the actual parameter may be written in the forms:

SYMBOL

SYMBOL ± CONSTANT

Parameters representing central memory locations may be written in the LITERAL forms:

(SYMBOL)

(SYMBOL ± CONSTANT)

Parameters representing numbers per se may be written in the forms:

CONSTANT

SYMBOL

SYMBOL ± CONSTANT

Parameters representing a file or program name must be written in the form:

SYMBOL

The following is an explanation of certain letters, terms, and phrases used in connection with macros:

A — Symbolic address in CP or PP memory which contains the CM address of the first word of the requested block assigned by the system or which contains the CM address, as specified by the programmer, of the first word of the block in memory to be released to the system. If the macro is used in a PP program, the CM address is absolute; but if used in a CP program, the address is relative.

BA — Symbolic address in CM or PP memory which contains the beginning address of the central memory buffer area.

C — Conversion mode

Card operations:

C = blank or 0 — no conversion (binary image)

1 — Hollerith to display code for read; display code to Hollerith for punch

2 – Hollerith to BCD for read; BCD to Hollerith for punch

Magnetic tape:

C = blank or 0 – no conversion
1 – BCD to display code
2 – display code to BCD

Printer:

C = blank or 0 – no conversion
2 – display code to BCD

D	Physical number (or PP symbolic address of number) of the I/O channel requested or released (this applies only to macros used in ASPER programs).
EA	Symbolic address in CM or PP memory which contains the ending address + 1 of the central memory buffer area.
K	Number (or CM or PP symbolic address of number) of logical tape records.
L	Number (or CM or PP symbolic address of number) of 60-bit words in the longest record in the file identified by NAME.
N	Equipment logical number (or CM or PP symbolic address of number), i.e., 1, 2 . . . M for M total units of equipment type in the system.
NAME	Symbolic name uniquely identifying the disk logical file being referenced.
NW	Total number (or CM or PP symbolic address of number) of central memory words requested or released.
P	Logical record number (or CM or PP symbolic address of number) in disk file to start read or write.
R	Maximum number (a CM or PP symbolic address of number) of logical records into which the disk file may be segmented.
RL	Record Length

Card operations: total number (or CM or PP symbolic address of number) of leftmost 5 columns (binary image) or 10-character fields (coded mode) of the card. For BCD or DPC conversion mode, each 60-bit word contains the 6-bit characters. For binary image, each 60-bit word contains 5 columns.

Console operations: total number (or CM or PP symbolic address of number) of characters in the message to be transmitted.

Magnetic tape: number (or CM or PP symbolic address of number) of 60-bit words per tape record.

Printer: number (or CM or PP symbolic address of number) of 10-character words per line to print.

S	Symbolic address in CM or PP memory which contains the address for the status response word from the PP I/O routine. Each macro request requires a location in central memory be reserved and identified. The PP I/O routine (or the system if the macro is in an ASPER program) reports to this location the status of the requested operation. The parameter for the status response word in a macro must be the same as that used in assigning the central memory location.
SYMBOL	Program overlay: name of overlay region to be loaded.

System actions: name of PP program defined by ASPER pseudo operation.

Wait check: name of transfer location of abort is indicated by the status response word.

T Display character size:

T = blank or 0 – 64 char./line
1 – 32 char./line
2 – 16 char./line
3 – plot mode

TAG Identification number $\leq$ 18 bits (or CM or PP symbolic address of number) of message to be displayed.

W A W appended to the opcode of a macro indicates a "wait for reply." if the W is not used (buffered mode), the CP or PP program may continue processing while the requested I/O operation is being performed. However, the program must do its own checking on the progress of the request by means of the WAI (Wait Check) macro. If the request is in process, the status response word is positive and nonzero; if the request is completed, the word is zero; if the request is aborted, the word is negative.

When the W is appended to the macro (non-buffered mode) and the requested operation can be performed, the action taken depends on whether the macro is in a CP or PP program. If in a CP program, control is turned oved to the operating system and the CP program delays until the status response word is zero (completed) or negative (aborted), at whieh time control is given back to the program. If the macro is in an ASPER program, the routine delays until status response word is zero (completed) or negative (aborted).

In both modes, the next in-line instructions is executed if the requested operation is successful.

4.1 MAGNETIC TAPE OPERATIONS

Opcode	Address Field	Remarks	
RQT<u>W</u>	N, S	Request tape assignment from system.	Wait if W used.
DRT<u>W</u>	N, S	Release tape back to system.	Wait if W used.
SFF<u>W</u>	N, S	Search file mark forward.	Wait if W used.
SFB<u>W</u>	N, S	Search file mark backward.	Wait if W used.
WFM<u>W</u>	N, S	Write file mark.	Wait if W used.
RWL<u>W</u>	N, S	Rewind tape to load point.	Wait if W used.
RWU<u>W</u>	N, S	Rewind tape for unload.	Wait if W used.
FSP<u>W</u>	N, S, K	Forespace	Wait if W used.
BSP<u>W</u>	N, S, K	Backspace	Wait if W used.
RFC<u>W</u>	N, S, BA, EA, RL, C	Read tape forward coded mode.	Wait if W used.
RFB<u>W</u>	N, S, BA, EA, RL, C	Read tape forward binary mode.	Wait if W used.
WRC<u>W</u>	N, S, BA, EA, RL, C	Write tape coded mode.	Wait if W used.
WRB<u>W</u>	N, S, BA, EA, RL, C	Write tape binary mode.	Wait if W used.

N = Magnetic tape logical unit number; 1,2, . . . M for M tape units in the system.

S = Location containing central memory address for status response code from System PP I/O routine.

K = Number of logical tape records.

BA = Location containing beginning address of buffer area in central memory.

EA = Location containing ending address +1 of buffer area in central memory.

RL = Number of 60-bit words per tape record.

C = Conversion mode.

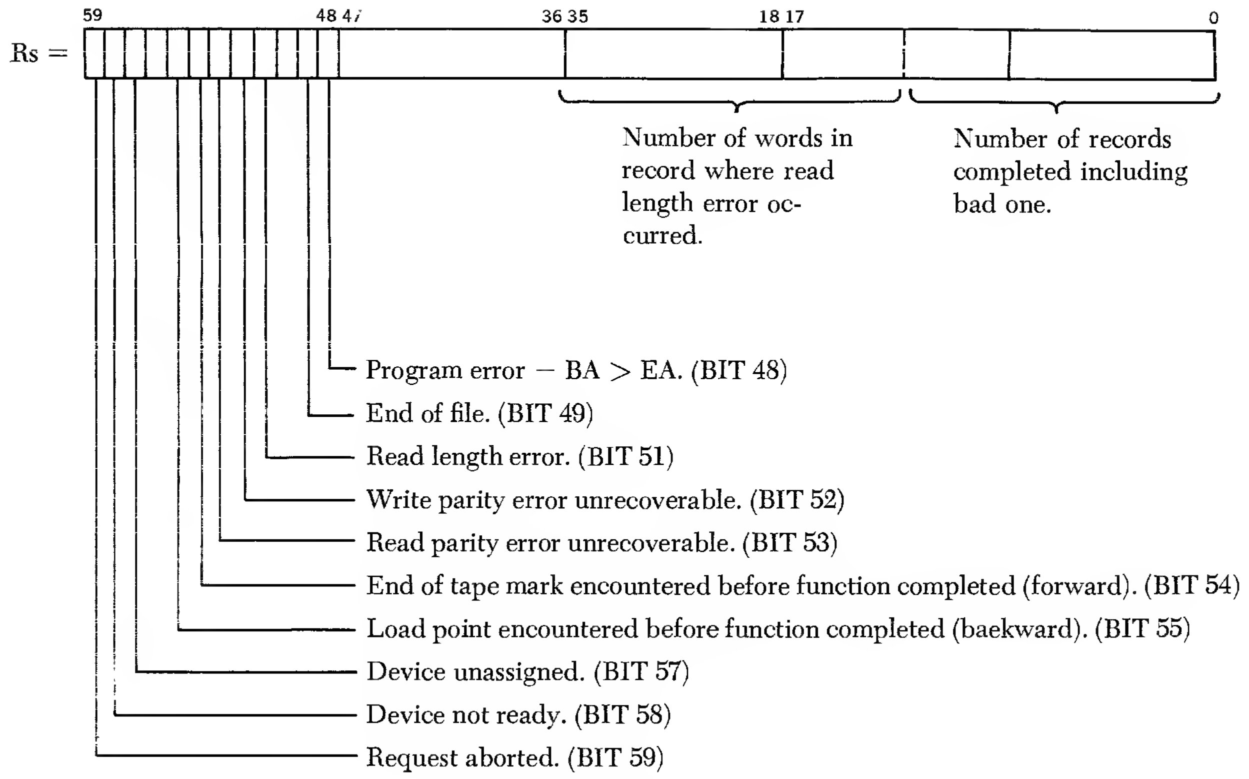
Blank or 0 — No conversion.
1 — BCD to display code.
2 — Display code to BCD.

STATUS RESPONSE CODES — positioned as per address S.

Rs = 0 Request completed with no trouble.

Rs — 1 Request in process.

Rs < 0 Request aborted. Reason given in bits 58-48.


Rs =
59
48 47
36 35
18 17
0
Number of words in record where read length error occurred.
Number of records completed including bad one.
Program error — BA > EA. (BIT 48)
End of file. (BIT 49)
Read length error. (BIT 51)
Write parity error unrecoverable. (BIT 52)
Read parity error unrecoverable. (BIT 53)
End of tape mark encountered before function completed (forward). (BIT 54)
Load point encountered before function completed (baekward). (BIT 55)
Device unassigned. (BIT 57)
Device not ready. (BIT 58)
Request aborted. (BIT 59)

where: 1 implies the condition exists.
0 implies the condition does not exist.

4.2 DISK TRANSFERS

Provision is made in the operating system for the programmer to read and write scratch data to and from disk storage units. Data are usually broken up into related blocks called files. The files, in turn, are segmented into the blocks of data that are transmitted at one time. These are called logical records. For most efficient utilization of disk storage, logical records contain a minimum of 512 central memory words. A file is defined by an instruction or statement which specifies the number of 60-bit words in the longest record, the maximum number of logical records into which the file is to be segmented, and the symbolic name by which the file is identified. The actual data transmission is accomplished through the use of the following macro operators.

Opcode	Address Field	Remarks	
RDH$\underline{W}$	N, S, BA, EA, NAME, P	Read record and hold data on disk.	Wait If W used.
RDR$\underline{W}$	N, S, BA, EA, NAME, P	Read record and release data on disk.	Wait if W used.
WRD$\underline{W}$	N, S, BA, EA, NAME, P	Write record on disk.	Wait if W used.

N = Disk logical unit number; 1,2, . . . M for M disk units in the system.

S = Location containing central memory address for status response code from System PP I/O routine.

BA = Location containing beginning address of buffer area in central memory.

EA = Location containing ending address +1 of buffer area in central memory.

NAME = Symbolic name to identify disk logical file to be referenced.

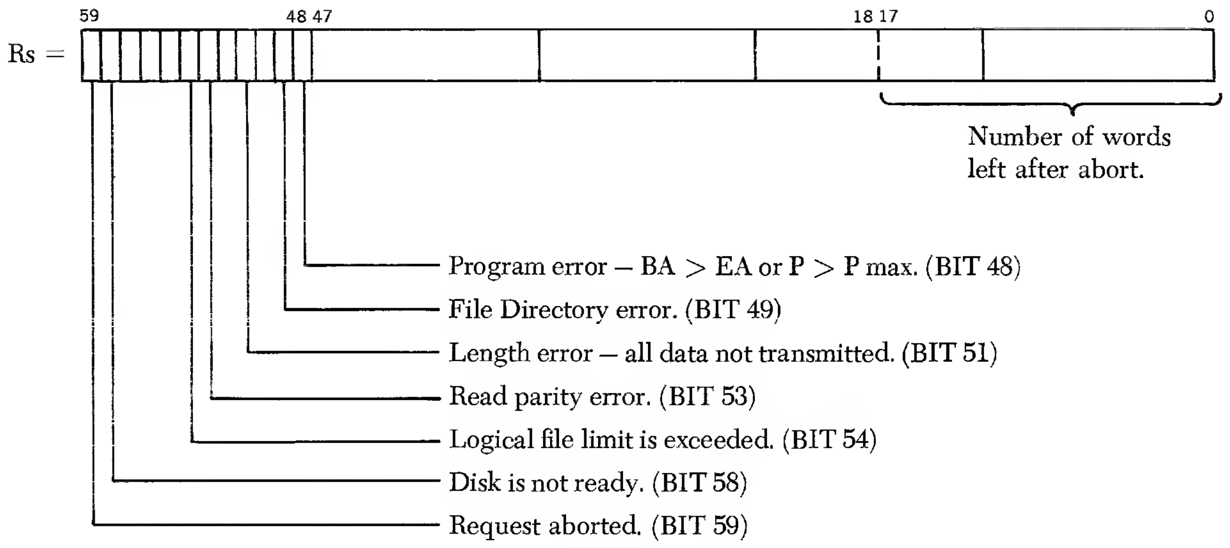
P = Logical record number used to identify record read from disk or written onto disk.

STATUS RESPONSE CODES – positioned as per address S.

Rs = 0 Request is completed with no trouble.

Rs = 1 Request is in process.

Rs < 0 Request aborted. Reason given in bits 58-48.



59
48 47
18 17
0
Rs =
Number of words left after abort.
Program error — BA > EA or P > P max. (BIT 48)
File Directory error. (BIT 49)
Length error — all data not transmitted. (BIT 51)
Read parity error. (BIT 53)
Logical file limit is exceeded. (BIT 54)
Disk is not ready. (BIT 58)
Request aborted. (BIT 59)

where: 1 implies the condition exists.

0 implies the condition does not exist.

4.3 PRINTER OPERATIONS

Opcode	Address Field	Remarks	
SSPW	N, S	Single space printer.	Wait if W is used.
DSPW	N, S	Double space printer.	Wait if W is used.
FC7W	N, S	Select Format Channel 7.	Wait if W is used.
FC8W	N, S	Select Format Channel 8.	Wait if W is used.
MC1W	N, S	Select Monitor Channel 1.	Wait if W is used.
MC2W	N, S	Select Monitor Channel 2.	Wait if W is used.
MC3W	N, S	Select Monitor Channel 3.	Wait if W is used.
MC4W	N, S	Select Monitor Channel 4.	Wait if W is used.
MC5W	N, S	Select Monitor Channel 5.	Wait if W is used.
MC6W	N, S	Select Monitor Channel 6.	Wait if W is used.
CMCW	N, S	Clear Monitor Channels 1 - 6.	Wait if W is used.
SPAW	N, S	Suppress space after next print.	Wait if W is used.
PRNW	N, S, BA, EA, RL, C	Print single line or multiple lines.*	Wait if W is used.

* If SPA is given preceding a multiple line print, it applies only to the first line.

N = Printer logical unit number; 1,2, . . . M for M printers in the system.

S = Location containing central memory address for status response code from System PP I/O routine.

BA = Location containing beginning address of buffer area in central memory.

EA = Location containing ending address +1 of buffer area in central memory.

RL = Number of 10 character words per line to print.

C = Conversion mode.

Blank or 0 — No conversion.

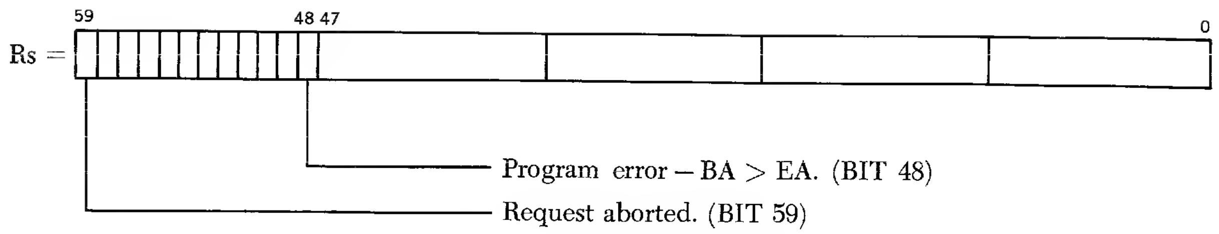
2 — Display code to BCD.

STATUS RESPONSE CODES — positioned as per address S.

Rs = 0 Request is completed with no trouble.

Rs = 1 Request is in process.

Rs < 0 Request aborted. Reason given in bits 58-48.



where: 1 implies the condition exists.
0 implies the condition does not exist.

4.4 CARD OPERATIONS

Opcode	Address Field	Remarks	
PCHW	N, S, BA, EA, RL, C	Punch cards.	Wait if W is used.
RDCW	N, S, BA, EA, RL, C	Read cards.	Wait if W is used.

N = Card reader or punch logical unit number; 1,2, . . . M for M readers or punches in the system.

S = Location containing central memory address for status response code from System PP I/O routine.

BA = Location containing beginning address of buffer area in central memory.

EA = Location containing ending address +1 of buffer area in central memory.

RL = Number of leftmost 10-character fields or 5 columns of the card.

C = Conversion mode.

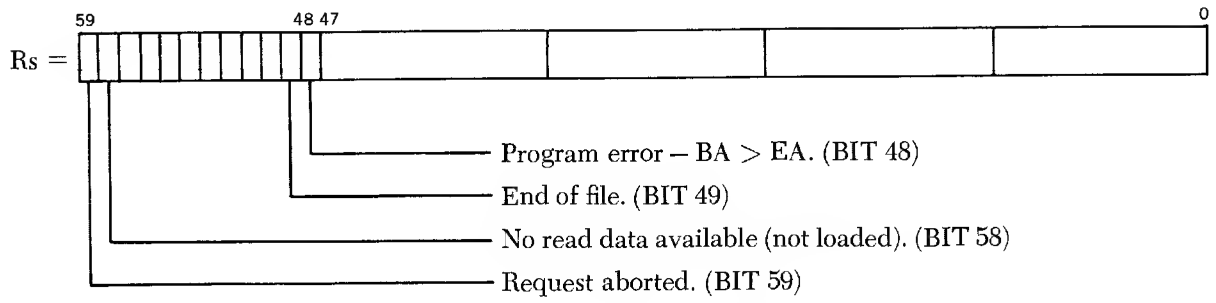
Blank or 0 – No conversion; i.e., binary image input/output.
1 – Hollerith to display code for read; display code to Hollerith for punch.
2 – Hollerith to BCD for read; BCD to Hollerith for punch.

STATUS RESPONSE CODES – positioned as per address S.

Rs = 0 Request is completed with no trouble.

Rs = 1 Request is in process.

Rs < 0 Request aborted. Rcason given in bits 58-48.



Rs =
59
48 47
0
Program error – BA > EA. (BIT 48)
End of file. (BIT 49)
No read data available (not loaded). (BIT 58)
Request aborted. (BIT 59)

where: 1 implies the condition exists.
0 implies the condition does not exist.

4.5 CONSOLE OPERATIONS

Request procedures are provided for CP or ASPER routines to display messages on the primary console right scope or either of the scopes on other consoles. The system provides a timing service for removal of displays after a certain exposure. However, the request procedure gives an option to override the system time limit on display. In this mode, it is assumed that the CP or ASPER routine will request a removal of the display as a result of console acknowledgment or internal decision.

Opcode	Address Field	Remarks	
DSRW	N, S, BA, EA, RL, TAG, T	Display on Right Scope for system time limit.	Wait if W is used.
DSLW	N, S, BA, EA, RL, TAG, T	Display on Left Scope for system time limit.	Wait if W is used.
DHRW	N, S, BA, EA, RL, TAG, T	Display on Right Scope and hold indefinitely	Wait if W is used.
DHLW	N, S, BA, EA, RL, TAG, T	Display on Left Scope and hold indefinitely	Wait if W is used.
RDPW	N, S, TAG	Remove display.	Wait if W is used.
RTYW	N, S, BA, EA, RL, TAG	Read console typewriter.	Wait if W is used.

N = Console logical unit number; 1,2,... M for M consoles in the system.

S = Location containing central memory address for status response codes from System PP I/O routine.

BA = Location containing beginning address of buffer area in central memory.

EA = Location containing ending address + 1 of buffer area in central memory.

RL = Total number of characters in the message to be transmitted.

TAG = Identification number $\leq$ 18 bits for display message.

T = Display character size.

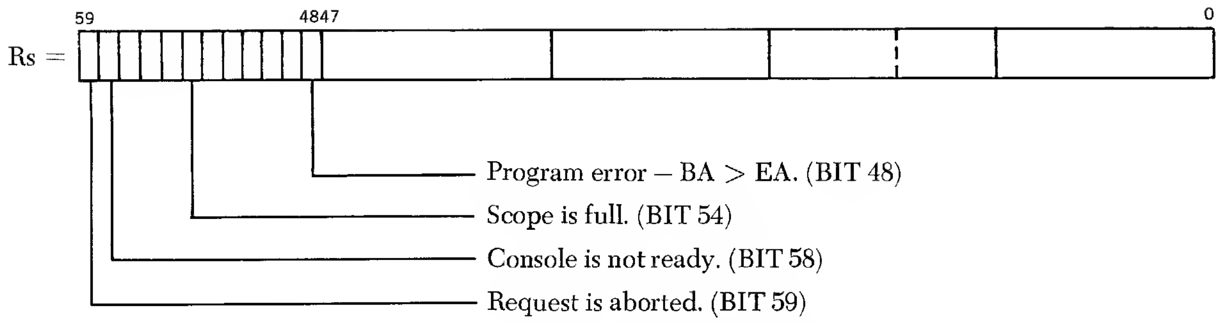
Blank or 0–64 characters/line.
1–32 characters/line.
2–16 characters/line.
3–plot mode.

STATUS RESPONSE CODES–positioned as per address S.

Rs = 0 Request is completed with no trouble.

Rs = 1 Request is in process.

Rs < 0 Request aborted. Reason given in bits 58-48.


59
4847
0
Rs =
Program error – BA > EA. (BIT 48)
Scope is full. (BIT 54)
Console is not ready. (BIT 58)
Request is aborted. (BIT 59)

where: 1 implies the condition exists
0 implies the condition does not exist.

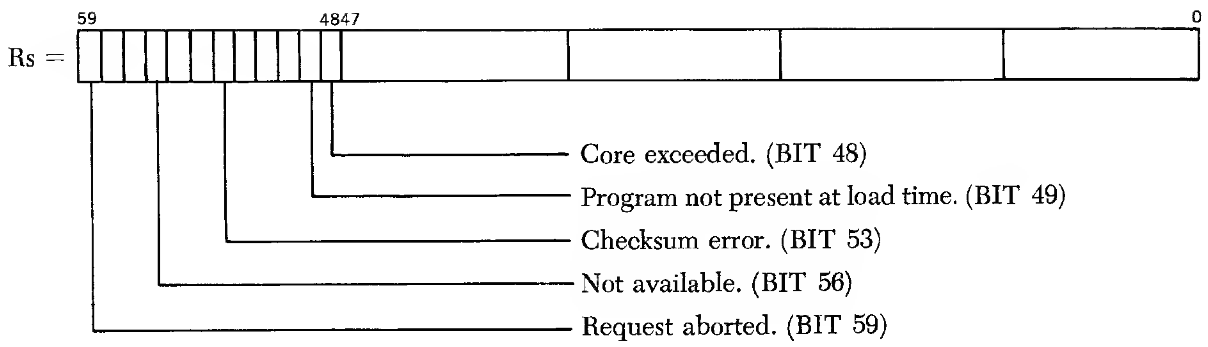
Opcode	Address Field	Remarks
TPPW	N, S, SYMBOL	Transfer program SYMBOL from CM to PP memory and begin execution with first ASPER instruction. Wait if W is used.
RQMW	NW, S, A	Request memory. Wait if W is used.
DRMW	NW, S, A	Release memory. Wait if W is used.
RQDW	N, S, L, NAME, R	Request disk space. Wait if W is used.
DRDW	N, S, NAME	Release disk space. Wait if W is used.
RQCW*	D, S	Request I/O channel. Wait if W is used.
DRCW*	D, S	Release I/O channel. Wait if W is used.
DRPP*	N, S	Release peripheral processor.

*These macros are used in ASPER programs only.

N =	Logical number of PP or disk unit.
S =	Location containing central memory address for status response code from System PP I/O routine.
D =	Physical number of the I/O channel requested.
R =	Maximum number of logical records into which the file may be segmented.
NW =	Total number of words.
L =	Number of 60-bit words in longest record.
A =	Location containing central memory address of the first word of block assigned by the system or released by the programmer.
NAME =	Symbolic name uniquely identifying the disk logical file being referenced.
SYMBOL =	Name of PP program defined by ASPER pseudo operation.

STATUS RESPONSE CODES—positioned as per address S.

Rs = 0	Request completed with no trouble.
Rs = 1	Request in process.
Rs < 0	Request aborted. Reason given in bits 58-48.


59
4847
0
Rs =
Core exceeded. (BIT 48)
Program not present at load time. (BIT 49)
Checksum error. (BIT 53)
Not available. (BIT 56)
Request aborted. (BIT 59)

4.7 CP PROGRAM OVERLAY

During initial loading, segmentation control cards are matched against subroutines present to assure overlay capability when called. Therefore, control is taken from the central memory program during the load and is only returned when the load is successful. The loading could fail because of an attempt to load a non-existent segment or subroutine. No status is required since success is necessary to regain control.

Opcode	Address Field	Remarks
LOAD	SYMBOL	Load segment SYMBOL.
LOAD	*SYMBOL*	Load segment SYMBOL and transfer control to indicated routine.

SYMBOL = Name of overlay region to be loaded.

4.8 PP PROGRAM OVERLAY

No execution takes place unless all SUBP's called in LOAD macros are present. During execution of the LOAD macro, control is kept in the macro and returned to the routine only upon successful completion of the load. Therefore, no status is provided.

Opcode	Address Field	Remarks
LOAD	SYMBOL	Load SUBP SYMBOL into PP memory

SYMBOL = Name of overlay region to be loaded.

4.9 WAIT CHECK

After a buffered operation is initiated, a Wait Check macro may be used to check status. The routine delays until the status response word is zero (completed) or negative (aborted). If it is zero, the next instruction in line is executed. If the status word is negative, the routine exits to the location specified by SYMBOL.

Opcode	Address Field	Remarks
WAI<u>W</u>	S, SYMBOL	Check status of S. Exit to SYMBOL if abort. Wait for reply if not ready and W is used.

S	=	Location containing central memory address for status response code from System PP I/O routine.
SYMBOL	=	Transfer location if an abort is indicated by the status response code.

4.10 DESCRIPTION OF MACRO INSTRUCTIONS

Backspace

    BSP   N, S, K

Backspaces K number of records on logical tape unit N.

Clear Monitor Channels 1-6

    CMC   N, S

Deselects monitor channels 1-6 on line printer N. This macro must be used before selecting another channel.

Display on Left Scope and Hold Indefinitely

    DHL   N, S, BA, EA, RL, TAG, T

Displays a message on the left scope of the console and holds the display indefinitely or until an RDP request is received. When displayed the message is accompanied by the 18-bit identifier, TAG. BA and EA contain the locations for the beginning and ending addresses of the buffer area storing the message to be displayed. Each CM word contains 10 consecutive display-coded characters of the message ordered from left to right in the word. The display character size is determined by T. RL specifies the number of characters to be displayed on each line on the scope and is limited by the character size chosen. The logical console number, N, indicates which console is to be used.

Display on Right Scope and Hold Indefinitely

    DHR   N, S, BA, EA, RL, TAG, T

Displays a message on the right scope of the console and holds the display indefinitely or until an RDP request is received. See macro DHL for further explanation of parameters.

Release Channel Back to System

    DRC   D, S

Releases the channel specified by D back to the system for general purpose use.

Release Disk Space Back to System

    DRD   N, S, NAME

Releases the file identified by NAME on the logical disk unit N.

Release Memory

    DRM   NW, S, A

Releases from the block of central memory words which the CP or PP has reserved the total number of words specified by NW beginning with the CM address given in A.

Release Tape Back to System

    DRT   N, S

Releases the logical tape unit specified by N for general system usage.

Display on Left Scope for System Time Limit

    DSL   N, S, BA, EA, RL, TAG, T

Displays a message on the left scope of the console for the length of time set by the system. See macro DHL for further explanation of parameters.

Double Space Printer

    DSP   N, S

Advances logical printer N two lines.

Display on Right Scope for System Time Limit

    DSR   N, S, BA, EA, RL, TAG, T

Displays a message on the right scope of the console for the length of time set by the system. See macro DHL for further explanation of parameters.

Select Format Channel 7

    FC7   N, S

Selects format channel 7 on logical printer unit N. This format channel advances the paper to a selected line.

Select Format Channel 8

    FC8   N, S

Selects format channel 8 on logical printer unit N. This format channel ejects the page to the top of the form.

Forespace

    FSP   N, S, K

Spaces forward K number of records on logical tape unit N.

Select Monitor Channel 1

MC1 N, S

Selects monitor channel 1 on logical printer unit N. The monitor channel contains predesigned line-space formats.

Select Monitor Channel 2

MC2 N, S

Select monitor channel 2 on logical printer unit N.

Select Monitor Channel 3

MC3 N, S

Select monitor channel 3 on logical printer unit N.

Select Monitor Channel 4

MC4 N, S

Select monitor channel 4 on logical printer unit N.

Select Monitor Channel 5

MC5 N, S

Select monitor channel 5 on logical printer unit N.

Select Monitor Channel 6

MC6 N, S

Select monitor channel 6 on logical printer unit N.

Punch Cards

PCH N, S, BA, EA, RL, C

Punches cards on logical unit N for the number of leftmost 5 columns (binary output, no conversion) or 10-character fields (coded mode) as given by RL. The conversion mode is specified by C. The card images are read out from central memory beginning at the address contained in location BA and ending at the address contained in location EA.

Print Single Line or Multiple Lines

PRN N, S, BA, EA, RL, C

Prints on logical unit N the number of 10-character words per line as given by RL in the conversion mode specified by C. RL may specify up to 12 or 14* words per line. The print image is stored in central memory beginning at the address contained in location BA and ending at the address contained in location EA.

Read Card

RDC N, S, BA, EA, RL, C

Reads cards on logical unit N for the number of leftmost 5 columns (binary input, no conversion) or 10-character fields (coded mode) as given by RL. The conversion mode is specified by C. The cards are read into central memory beginning at the address contained in location BA and ending at the address contained in location EA.

Read Record and Hold Data on Disk

RDH N, S, BA, EA, NAME, P

Reads the record identified by NAME commencing with record number P from logical disk unit N. BA and EA contain the locations for the beginning and ending addresses of the buffer into which the words are read. The data are held on disk for subsequent re-use.

Remove Display

RDP N, S, TAG

Erases from the scope at console N the display identified by TAG.

Read Record and Release Data on Disk

RDR N, S, BA, EA, NAME, P

Reads the record identified by NAME commencing with record number P from logical disk unit N. BA and EA contain the locations for the beginning and ending addresses of the buffer into which the macros are read. Once the data are in memory, the disk space is released for use by other programs.

Read Tape Forward, Binary Mode

RFB N, S, BA, EA, RL, C

Reads, in binary parity, the number of 60-bit words per tape record, RL, from logical tape unit N. Each 6-bit character is converted as specified by the conversion mode C. BA and EA contain the locations for the beginning and ending addresses of the buffer into which the words are read.

*for the 120 character/line 1612 printer and the 136 character/line 501 printer, respectively.

Read Tape Forward, Coded Mode

RFC N, S, BA, EA, RL, C

Reads, in BCD parity, the number of 60-bit words per tape record, RL, from logical tape unit N. Each 6-bit character is converted as specified by the conversion mode C. BA and EA contain the locations for the beginning and ending addresses of the buffer into which the words are read.

Request Channel

RQC D, S

Requests the channel specified by D for the exclusive use of the requesting program.

Request Disk Space

RQD N, S, L, NAME, R

Reserves on logical disk unit N the file identified by NAME which has L number of 60-bit words in its longest record. R specifies the maximum number of logical records into which the file may be segmented. The parameters N, L, and R must be numbers, where $N \leq 16_{10}$, $L \leq 2^{18}$, and $R \leq 4000_{10}$. NAME must be unique within the routine.

Request Memory Space

RQM NW, S, A

Reserves in central memory the total number of words specified by NW. The system sets A to the location containing the address of the first word of the assigned block in central memory.

Request Tape Assignment from System

RQT N, S

Requests logical tape unit N for the exclusive use of a program.

Read Console Typewriter

RTY N, S, BA, EA, RL, TAG

Reads and identifies a message with the identification number, TAG, typed on the typewriter at logical console unit N. Transmits RL number of characters to a buffer area in central memory beginning at the address contained in location BA and ending at the address contained in location EA.

Rewind Tape to Load Point

RWL N, S

Rewinds logical tape unit N to the physical load point on the tape.

Rewind Tape for Unload

RWU N, S

Rewinds logical tape unit N so that the tape may be dismounted.

Search File Mark Backward

SFB N, S

Searches the tape on logical unit N one record at a time back towards the load point until a file mark is passed over. When the mark is found, the tape is positioned on the load-point side of the file mark. If none is found, the macro is equivalent to RWL.

Search File Mark Forward

SFF N, S

Searches the tape on logical unit N one record at a time from the current position forward until a file mark is passed over. When the mark is found, the tape is positioned on the side of the file mark away from the load point. If no mark is found, the end of tape marker stops the search.

Suppress Space After Next Print

SPA N, S

Suppresses on logical printer N the automatic advance after the next line printed with a PRN macro.

Single-Space Printer

SSP N, S

Advances logical printer N one line.

Transfer PP Program and Begin Execution

TPP N, S, Symbol

Produces a calling sequence to the PP loader which, during execution, transfers PP program SYMBOL from central memory to logical peripheral processor N and begins execution with the first ASPER instruction. This macro is used to load an ASPER program into a PP from CM at execute time. The load begins at the first binary card and continues until the loader encounters another ASPER header card, a SUBP header card, or a terminate card. Execution begins at the first ASPER instruction defined under an ORGR pseudo code.

The TPP call from a CM program can load any PP in the system. However, the TPP call by a PP program can load any other PP in the system but cannot load itself.

Wait Check

```
    WAI     S, SYMBOL
```

Checks the status response word of other macros during a buffered operation. If the operation has been aborted, the WAI macro exits to the address specified by SYMBOL. If not, the next instruction, in line, is executed.

Write File Mark

```
    WFM     N, S
```

Writes an end of file mark on the tape on logical unit N.

Write Tape, Binary Mode

```
    WRB     N, S, BA, EA, RL, C
```

Writes, in binary parity, the data between BA and EA in records of RL 60-bit words each onto logical tape unit N. Each 6-bit character transferred is converted as requested by the conversion mode C. The words are written from a buffer area in central memory beginning at the address contained in location BA and ending at the address contained in location EA. If the conversion mode is 0, a straight binary output is expected.

Write Tape, Coded Mode

```
    WRC     N, S, BA, EA, RL, C
```

Writes, in BCD parity, the data between BA and EA in records of RL 60-bit words each onto logical tape unit N. Each 6-bit character transferred is converted as requested by the conversion mode C. The words are written from a buffer area in central memory beginning at the address contained in location BA and ending at the address contained in location EA.

Write Record on Disk

```
    WRD     N, S, BA, EA, NAME, P
```

Writes on logical disk N the logical record identified by NAME starting with reeord number P. The words are written without code translation from a buffer area in central memory beginning at the address contained in location BA and ending at the address contained in location EA.

Release Peripheral Processor

```
    DRPP    N, S
```

Returns the PP, logical unit N, to the system for general purpose use. This macro must be the final instruction executed before the program completes.

Load Segment

```
    LOAD    SYMBOL
```

Loads the subroutine SYMBOL into CP or PP memory. SYMBOL is a subroutine defined by a pseudo opcode. When asterisks enclose SYMBOL, the control is transferred to the indicated routine.

5. UTILITY ROUTINES

The system library on the disk contains several standard utility routines that may be put into operation by control cards. These routines perform common input/output operations and, once set up and initiated by the operating system, are executed off-line. Routines in this group include:

Card to Punch
Card to Print
Card to Tape
Tape Comparison
Tape to Card
Tape to Print
Tape to Tape

5.1 CARD TO PUNCH

The card-to-punch routine reads card data into the peripheral processor's memory from the card reader and punches out identical information on the card punch. Any code may be punched in the cards being read.

Required parameters:

- Channel number of card reader
- Synchronizer number of card reader
- Unit number of card reader
- Channel number of card punch
- Synchronizer number of card punch
- Unit number of card punch

Error halt conditions:

- Card reader not ready
- Card punch not ready

5.2 CARD TO PRINT

The card-to-print routine reads Hollerith coded data into the peripheral processor's memory from the card reader, converts the data to BCD code, and prints the data on the line printer.

Required parameters:

- Channel number of card reader
- Synchronizer of card reader
- Unit number of card reader
- Channel number of line printer
- Synchronizer of line printer
- Unit number of line printer
- Conversion code
  - 0 = No conversion
  - 2 = Convert from Hollerith code

Error halt conditions:

- Card reader not ready
- Line printer not ready

5.3 CARD TO TAPE

The card-to-tape routine reads Hollerith coded data into the peripheral processor's memory from the card reader, converts the data to BCD or display code, arranges the data into blocks, and writes the blocks on magnetic tape (607 or 626).

Required parameters:

- Channel number of card reader
- Synchronizer number of card reader
- Unit number of card reader
- Channel number of magnetic tape
- Synchronizer number of magnetic tape
- Unit number of magnetic tape
- Input record length (number of columns per card to be read)
- Output record length (number of input records per block)
- Type of padding
  - 0 = Follow last record with zeros
  - 9 = Follow last record with nines
- Conversion code
  - 0 = No conversion
  - 1 = Convert Hollerith to BCD code
  - 2 = Convert Hollerith to display code

Error halt conditions:

- Card reader not ready
- Tape unit not ready
- Read parity error unrecoverable
- Write parity error unrecoverable
- End of tape before function completed

5.4 TAPE COMPARISON

The magnetic tape comparison routine reads data into the peripheral processor's memory from the two magnetic tape units, compares the data frame for frame, and exits when one of the following conditions is detected: unequal comparison, five consecutive file marks, or end of tape. Either 607 or 626 tapes may be compared.

Required parameters:

- Channel numbers for both tapes
- Synchronizer numbers for both tapes
- Unit numbers for both tapes

Error halt conditions:

- Tape units not ready
- Tapes not at load point
- Read parity error unrecoverable
- End of tape before function completed
- Records not equal

5.5 TAPE TO CARD

The tape-to-card routine reads binary or BCD data into the peripheral processor's memory from magnetic tape (607 or 626), converts the data to Hollerith code, and punches the data into cards.

Required parameters:

- Channel number for magnetic tape
- Synchronizer number for magnetic tape
- Unit number for magnetic tape
- Channel number for card punch
- Synchronizer number for card punch
- Unit number for card punch
- Tape mode
  - 0 = Binary
  - 1 = BCD
- Tape record length (frames per record)
- Number of files to be processed
- Number of files to be skipped (before starting)
- Conversion code
  - 0 = No conversion
  - 1 = Convert BCD to Hollerith
  - 2 = Convert binary to Hollerith

Error halt conditions:

- Tape unit not ready
- Card punch not ready
- Tape not at load point
- Read length error
- Read parity error unrecoverable

5.6 TAPE TO PRINT

The tape-to-print routine reads BCD, binary, or display coded data into the peripheral processor's memory from magnetic tape (607 or 626), converts the data if necessary to BCD code, and prints the data on the line printer. Tape records may be any length and any number of characters per line may be printed.

Required parameters:

- Channel number of magnetic tape
- Synchronizer number of magnetic tape
- Unit number of magnetic tape
- Channel number of line printer
- Synchronizer of line printer
- Unit number of line printer
- Tape mode
  - 0 = Binary
  - 1 = BCD
- Tape record length (12-bit words per record)
- Number of characters per line
- Number of file to be processed
- Number of files to be skipped (before starting)
- Conversion code
  - 0 = No conversion
  - 1 = Convert from binary
  - 2 = Convert from display code

Error halt conditions:

- Tape unit not ready
- Line printer not ready
- Tape not at load point
- Read length error
- Read parity error unrecoverable

5.7 TAPE TO TAPE

The tape-to-tape routine reads data into the peripheral processor's memory from one tape and writes identical data on another tape. This is merely a tape duplication routine and will duplicate any number of files on either the 607 or 626 tape unit.

Required parameters:

- Channel numbers for both tapes
- Synchronizer numbers for both tapes
- Unit numbers for both tapes

Tape mode of input tape

0 = Binary

1 = BCD

Input record length (maximum length)

Number of files to process

Error halt conditions:

Tape units not ready
Tapes not at load point
Tape write lockout
Read parity error unrecoverable
Write parity error unrecoverable
End of tape before function completed

APPENDIX

1. CHARACTERISTICS SUMMARY

SYSTEM

Large-scale, general-purpose computer system

11 independent computers
- 1 central processor (60-bit)
- 10 peripheral and control processors (12-bit)
- Central memory (60-bit)
- Display console and keyboard

System communicates with a variety of external equipment
- Disk files
- Magnetic tapes
- Card equipment
- Printers

Central memory common to the 11 computers

Central memory storage
- 131,072 words (60-bit)
- Major cycle = 1000 ns*
- Minor cycle = 100 ns
- Memory organized in 32 banks of 4096 words
- Multiphase

Central processor instructions
- Arithmetic, logical, indexing, branch

Peripheral and control processor instructions
- Logical, input-output, access to central processor and central memory

Each peripheral and control processor has 12-bit, 4096-word memory

Solid-state system
- Transistor logic

*ns = nanoseconds

CENTRAL PROCESSOR

10 arithmetic and logical units

Add	Shift
Multiply	Branch
Multiply	Boolean
Divide	Increment
Long add	Increment

24 operating registers for functional units
- 8 operand (60-bit)
- 8 address (18-bit)
- 8 increment (18-bit)

8 transistor registers (60-bit) hold 32 instructions (15-bit) or 16 instructions (30-bit) or combination of two for servicing functional units

Floating point add — 4 minor cycles

Floating point multiply — 10 minor cycles

Floating point divide — 29 minor cycles

Floating point arithmetic
- Single and double precision
- Optional rounding and normalizing
- Format
  - Integer coefficient — 48 bits
  - Biased exponent — 11 bits (2^{10})
  - Coefficient sign — 1 bit

Fixed point arithmetic (subset of floating point arithmetic)
- Full 60-bit add-subtract

Controlled and started by peripheral and control processors

Addresses in central memory relative

PERIPHERAL AND CONTROL PROCESSORS

10 identical processors (characteristics as listed are per processor except as noted)

4096 word magnetic core memory (12-bit)
- Random access, coincident — current
- Major cycle — 1000 ns
- Minor cycle — 100 ns

12 input/output channels
- All channels common to all processors
- Maximum transfer rate per channel — one word/major cycle
- All 12 channels may be active simultaneously
- All channels 12-bit bi-directional

Real-time clock (period = 4096 major cycles)

Instructions
- Add/Subtract
- Logical
- Branch
- Input/Output
- Central processor access
- Central memory access

Average instruction execution time = two major cycles

Indirect addressing

Indexed addressing

CENTRAL MEMORY

131,072 words

60-bit words

Memory organizing in 32 logically independent banks of 4096 words with corresponding multi-phasing of banks

Random access, coincident — current, magnetic core

One major cycle for read-write

Maximum memory reference rate to all banks
 one address/minor cycle

Maximum rate of data flow to/from memory
 one word/minor cycle

DISPLAY CONSOLE

Two display tubes

Modes
- Character
- Dot

Character size
- Large — 16 characters/line
- Medium — 32 characters/line
- Small — 64 characters/line

Characters
- 26 alphabetie
- 10 numeric
- 11 special

2. SYSTEM MACROS

RQTW	Request tape assignment from system.
DRTW	Release tape back to system.
SFFW	Search file mark forward.
SFBW	Search file mark backward.
WFMW	Write file mark.
RWLW	Rewind tape to load point.
RWUW	Rewind tape for unload.
FSPW	Forespace.
BSPW	Backspace.
RFCW	Read tape forward coded mode.
RFBW	Read tape forward binary mode.
WRCW	Write tape coded mode.
WRBW	Write tape binary mode.
RDHW	Read record and hold data on disk.
RDRW	Read record and release data on disk.
WRDW	Write record on disk.
SSPW	Single space printer.
DSPW	Double space printer.
FC7W	Select Format Channel 7.
FC8W	Select Format Channel 8.
MC1W	Select Monitor Channel 1.
MC2W	Select Monitor Channel 2.
MC3W	Select Monitor Channel 3.
MC4W	Select Monitor Channel 4.
MC5W	Select Monitor Channel 5.
MC6W	Select Monitor Channel 6.
CMCW	Clear Monitor Channels 1–6.
SPAW	Suppress space after next print.
PRNW	Print single line or multiple lines.
PCHW	Punch cards.
RDCW	Read cards.
DSRW	Display on Right Scope for system time limit.
DSLW	Display on Left Scope for system time limit.
DHRW	Display on Right Scope and hold indefinitely.
DHLW	Display on Left Scope and hold indefinitely.
RDPW	Remove display.
RTYW	Read console typewriter.
WAIW	Check status word.
TPPW	Transfer program SYMBOL from CM to PP memory and begin execution with first ASPER instruction.
LOAD	Load segment SYMBOL.
RQMW	Request memory.
DRMW	Release memory.
RQDW	Request disk.
DRDW	Release disk space.
RQCW	Request I/O channel.
DRCW	Release I/O channel.
DRPP	Release peripheral processor.

3. PERIPHERAL PROCESSOR OPERATION CODES

Octal Opcode	Mnemonic	Address	Comments
00	PSN		. Pass
01	LJM	m d	. Long jump to m + (d)
02	RJM	m d	. Return jump to m + (d)
03	UJN	d	. Unconditional jump d
04	ZJN	d	. Zero jump d
05	NJN	d	. Nonzero jump d
06	PJN	d	. Plus jump d
07	MJN	d	. Minus jump d
10	SHN	d	. Shift d
11	LMN	d	. Logical difference d
12	LPN	d	. Logical product d
13	SCN	d	. Selective clear d
14	LDN	d	. Load d
15	LCN	d	. Load complement d
16	ADN	d	. Add d
17	SBN	d	. Subtract d
20	LDC	dm	. Load dm
21	ADC	dm	. Add dm
22	LPC	dm	. Logical product dm
23	LMC	dm	. Logical difference dm
24	PSN		. Pass
25	PSN		. Pass
26	EXN		. Exchange jump
27	RPN		. Read program address
30	LDD	d	. Load (d)
31	ADD	d	. Add (d)
32	SBD	d	. Subtract (d)
33	LMD	d	. Logical difference (d)
34	STD	d	. Store (d)
35	RAD	d	. Replace add (d)
36	AOD	d	. Replace add one (d)
37	SOD	d	. Replace subtract one (d)
40	LDI	d	. Load ((d))
41	ADI	d	. Add ((d))
42	SBI	d	. Subtract ((d))
43	LMI	d	. Logical difference ((d))
44	STI	d	. Store ((d))
45	RAI	d	. Replace add ((d))
46	AOI	d	. Replace add one ((d))

Octal Opcode	Mnemonic	Address	Comments
47	SOI	d	. Replace subtract one ((d))
50	LDM	m d	. Load (m + (d))
51	ADM	m d	. Add (m + (d))
52	SBM	m d	. Subtract (m + (d))
53	LMM	m d	. Logical difference (m + (d))
54	STM	m d	. Store (m + (d))
55	RAM	m d	. Replace add (m + (d))
56	AOM	m d	. Replace add one (m + (d))
57	SOM	m d	. Replace subtract one (m + (d))
60	CRD	d	. Central read from (A) to d
61	CRM	m d	. Central read (d) words from (A) to m
62	CWD	d	. Central write to (A) from d
63	CWM	m d	. Central write (d) words to (A) from m
64	AJM	m d	. Jump to m if channel d active
65	IJM	m d	. Jump to m if channel d inactive
66	FJM	m d	. Jump to m if channel d full
67	EJM	m d	. Jump to m if channel d empty
70	IAN	d	. Input to A from channel d
71	IAM	m d	. Input (A) words to m from channel d
72	OAN	d	. Output from A on channel d
73	OAM	m d	. Output (A) words from m on channel d
74	ACN	d	. Activate channel d
75	DCN	d	. Disconnect channel d
76	FAN	d	. Function (A) on channel d
77	FNC	m d	. Function m on channel d

FOOTNOTE TO APPENDIX 3

Notation	Interpretation
d	Implies d itself
(d)	Implies the contents of d
((d))	Implies the contents of the location specified by d
m	Implies m itself used as an address
m + (d)	The contents of d are added to m to form an operand (jump address)
(m + (d))	The contents of d are added to m to form the address of the operand
dm	Implies an 18-bit quantity with d as the upper 6 bits and m as the lower 12 bits

4. CENTRAL PROCESSOR OPERATION CODES

Octal Opcode	Mnemonic	Address	Comments
			. **BRANCH UNIT**
00	PS		. Program stop
01	RJ	K	. Return jump to K
02	JP	Bi + K	. Jump to Bi + K
030	ZR	Xj K	. Jump to K if Xj = 0
031	NZ	Xj K	. Jump to K if Xj ≠ 0
032	PL	Xj K	. Jump to K if Xj = plus (positive)
033	NG	Xj K	. Jump to K if Xj = negative
034	IR	Xj K	. Jump to K if Xj is in range
035	OR	Xj K	. Jump to K if Xj is out of range
036	DF	Xj K	. Jump to K if Xj is definite
037	ID	Xj K	. Jump to K if Xj is indefinite
04	EQ	Bi Bj K	. Jump to K if Bi = Bj
04	ZR	Bi K	. Jump to K if Bi = B0
05	NE	Bi Bj K	. Jump to K if Bi ≠ Bj
05	NZ	Bi K	. Jump to K if Bi ≠ B0
06	GE	Bi Bj K	. Jump to K if Bi ≥ Bj
06	PL	Bi K	. Jump to K if Bi ≥ B0
07	LT	Bi Bj K	. Jump to K if Bi < Bj
07	NG	Bi K	. Jump to K if Bi < B0
			. **BOOLEAN UNIT**
10	BXi	Xj	. Transmit Xj to Xi
11	BXi	Xj*Xk	. Logical Product of Xj & Xk to Xi
12	BXi	Xj + Xk	. Logical sum of Xj & Xk to Xi
13	BXi	Xj − Xk	. Logical difference of Xj & Xk to Xi
14	BXi	− Xk	. Transmit the comp. of Xk to Xi
15	BXi	− Xk*Xj	. Logical product of Xj & Xk comp. to Xi
16	BXi	− Xk + Xj	. Logical sum of Xj & Xk comp. to Xi
17	BXi	− Xk − Xj	. Logical difference of Xj & Xk comp. to Xi
			. **SHIFT UNIT**
20	LXi	jk	. Left shift Xi, jk places
21	AXi	jk	. Arithmetic right shift Xi, jk places
22	LXi	Bj Xk	. Left shift Xk nominally Bj places to Xi
23	AXi	Bj Xk	. Arithmetic right shift Xk nominally Bj places to Xi
24	NXi	Bj Xk	. Normalize Xk in Xi and Bj
25	ZXi	Bj Xk	. Round and normalize Xk in Xi and Bj
26	UXi	Bj Xk	. Unpack Xk to Xi and Bj
27	PXi	Bj Xk	. Pack Xi from Xk and Bj
43	MXi	jk	. Form mask in Xi, jk bits

Octal Opcode	Mnemonic	Address	Comments
			. **ADD UNIT**
30	FXi	Xj + Xk	. Floating sum of Xj and Xk to Xi
31	FXi	Xj − Xk	. Floating difference Xj and Xk to Xi
32	DXi	Xj + Xk	. Floating DP sum of Xj and Xk to Xi
33	DXi	Xj − Xk	. Floating DP difference of Xj and Xk to Xi
34	RXi	Xj + Xk	. Round floating sum of Xj and Xk to Xi
35	RXi	Xj − Xk	. Round floating difference of Xj and Xk to Xi
			. **LONG ADD UNIT**
36	IXi	Xj + Xk	. Integer sum of Xj and Xk to Xi
37	IXi	Xj − Xk	. Integer difference of Xj and Xk to Xi
			. **MULTIPLY UNIT**
40	FXi	Xj * Xk	. Floating product of Xj and Xk to Xi
41	RXi	Xj * Xk	. Round floating product of Xj & Xk to Xi
42	DXi	Xj * Xk	. Floating DP product of Xj & Xk to Xi
			. **DIVIDE UNIT**
44	FXi	Xj / Xk	. Floating divide Xj by Xk to Xi
45	RXi	Xj / Xk	. Round floating divide Xj by Xk to Xi
46	NO		. No operation
47	CXi	Xk	. Count the number of 1's in Xk to Xi
			. **INCREMENT UNIT**
50	SAi	Aj + K	. Set Ai to Aj + K
50	SAi	Aj − K	. Set Ai to Aj + comp. of K
51	SAi	Bj + K	. Set Ai to Bj + K
51	SAi	Bj − K	. Set Ai to Bj + comp. of K
52	SAi	Xj + K	. Set Ai to Xj + K
52	SAi	Xj − K	. Set Ai to Xj + comp. of K
53	SAi	Xj + Bk	. Set Ai to Xj + Bk
54	SAi	Aj + Bk	. Set Ai to Aj + Bk
55	SAi	Aj − Bk	. Set Ai to Aj − Bk
56	SAi	Bj + Bk	. Set Ai to Bj + Bk
57	SAi	Bj − Bk	. Set Ai to Bj − Bk
60	SBi	Aj + K	. Set Bi to Aj + K
60	SBi	Aj − K	. Set Bi to Aj + comp. of K
61	SBi	Bj + K	. Set Bi to Bj + K
61	SBi	Bj − K	. Set Bi to Bj + comp. of K
62	SBi	Xj + K	. Set Bi to Xj + K
62	SBi	Xj − K	. Set Bi to Xj + comp. of K
63	SBi	Xj + Bk	. Set Bi to Xj + Bk
64	SBi	Aj + Bk	. Set Bi to Aj + Bk
65	SBi	Aj − Bk	. Set Bi to Aj − Bk
66	SBi	Bj + Bk	. Set Bi to Bj + Bk
67	SBi	Bj − Bk	. Set Bi to Bj − Bk

Octal Opcode	Mnemonic	Address	Comments
70	SXi	Aj + K	. Set Xi to Aj + K
70	SXi	Aj − K	. Set Xi to Aj + comp. of K
71	SXi	Bj + K	. Set Xi to Bj + K
71	SXi	Bj − K	. Set Xi to Bj + comp. of K
72	SXi	Xj + K	. Set Xi to Xj + K
72	SXi	Xj − K	. Set Xi to Xj + comp. of K
73	SXi	Xj + Bk	. Set Xi to Xj + Bk
74	SXi	Aj + Bk	. Set Xi to Aj + Bk
75	SXi	Aj − Bk	. Set Xi to Aj − Bk
76	SXi	Bj + Bk	. Set Xi to Bj + Bk
77	SXi	Bj − Bk	. Set Xi to Bj − Bk

FOOTNOTE TO APPENDIX 4

A	One of eight address registers (18 bits)
B	One of eight index registers (18 bits) B0 is fixed and equal to zero
fm	Instruction code (6 bits)
i	Specifies which of eight designated registers (3 bits)
j	Specifies which of eight designated registers (3 bits)
jk	Constant, indicating number of shifts to be taken (6 bits)
k	Specifies which of eight designated registers (3 bits)
K	Constant, indicating branch destination or operand (18 bits)
X	One of eight operand registers (60 bits)

5. BINARY CARD FORMATS

There are several types of binary card formats that could appear in the job deck. Most of these are the result of assemblies or compilations. They are distinguished from one another by means of specific identification punches in column one. To make the definition of these punches easier, column one has been assumed to be a 12-bit word with each punch row in the column corresponding to a bit in a word; the 12 row corresponds to the high-order end of the word and the 9 row to the low-order end. As an example, Figure 15 shows the number 0005 punched into a card.

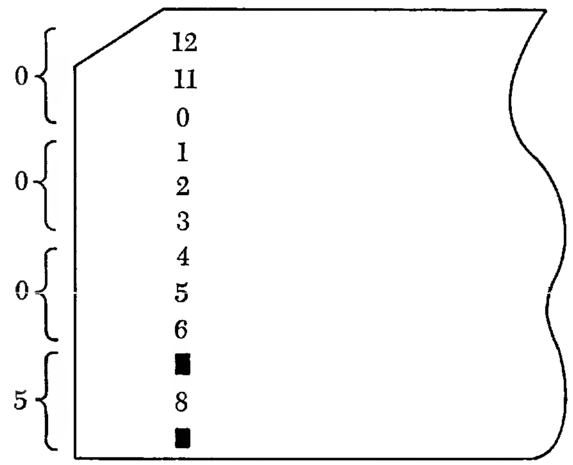


Figure 15. BINARY CARD FORMAT

Below is a table defining the card identification using the above convention. These entries can be logically "ored" with any other entry to convey a combination of meanings.

Card ID	*Definition*
0005	Binary card containing central memory words
0007	Binary card containing peripheral processor words
0010	Termination card (generated by source language END card)
0100	Card contains list of subroutines referenced by the routine
0200	Card contains subroutine names and entry points
0300	Card contains program name and entry point
0400	Card contains list of disk space requirements
0500	Card contains list of common definitions
1000	Checksum is to be ignored
2000	Card contains program instructions whose addresses are to be relocated (also contains a relocatable directory)
4000	Load address is to be relocated

Examples: In the examples below, card ID codes 2000, 4000, and either 0005 or 0007 are combined.

Card ID	*Definition*
6005	Binary card containing central memory instructions whose addresses may be relocated. The load address of the card may also be relocated. This is normal FORTRAN or ASCENT output.
6007	The same as above except the instructions are peripheral processor instructions. This is normal ASPER output.

CENTRAL PROCESSOR CARDS

Segment Control Cards

These cards define the way programs which are too large to fit into memory at one time are to be segmented and executed.

Column 1: CARD ID – This column identifies the type of segmentation card and must have one of the following codes:

Code	Description
0023	Basic Segment
0043	Normal Segment
0063	Continuation Card

Column 2: SEGMENT NAME – This is the name of the segment and may be from one to eight characters. Name is followed by a comma.

NAME OF PROGRAM, SUBROUTINE, or SEGMENT – The names of all programs, subroutines, or segments in SEGMENT NAME are listed here. Each name may have from one to eight characters and is followed by a comma. If more names than can be entered in one card are defined for a segment, they are entered in continuation cards.

Columns 76-80: DECK ID.

Common Areas Definitions

These cards reserve central memory for all common areas defined in the program and also reserve central memory required by the disk file.

Column 1: CARD ID – 12-bit identification code (0505).

Columns 2-3: LOAD ADDRESS – This is a card sequence number. Column two contains the high-order 12 bits of the number and column three contains the low-order 6 bits (rows 12-3).

Column 4: ENTRY COUNT – This is a 6-bit count (rows 4-9) of the number of common areas defined on this card.

Column 5: CHECK SUM – This is the sum of all columns except 5, 76-80.

Columns 6-10: Blank.

Columns 11-14: COMMON AREA NAME – This consists of up to eight BCD character codes starting in column 11.

Columns 15-16: LENGTH OF AREA – This is an 18 bit number which designates the number of central memory words to be reserved for the common area named. Column 15 contains the high-order 6 bits (rows 4-9) and column 16 contains the low order 12 bits.

Columns 17-75: OTHER ENTRIES.

Columns 76-80: DECK ID.

Common Area Data Card

This card contains data to be loaded into common areas.

Column 1: CARD ID – 12-bit identification code (0605).

Columns 2-3: LOAD ADDRESS – This is a common address of the area in which the data are to be loaded. Column two contains the high-order 12 bits, of the number and column three contains the low-order 6 bits (rows 12-3).

Column 4: WORD COUNT – This is a 6-bit count (rows 4-9) of the number of 60-bit data words contained in the card (up to 14 are allowed).

Column 5: CHECK SUM – This is the sum of all columns except 5, 76-80.

Columns 6-10: Ist DATA WORD – These columns contain the first 60-bit data word.

Columns 11-75: OTHER DATA WORDS.

Columns 76-80: DECK ID.

Disk Space Requirements

Disk space requirements cards define the disk files to be used by the central processor routine and reserve space in the programmer scratch area for the files.

Column 1: CARD ID – 12-bit identification code (0405).

Columns 2-3:	LOAD ADDRESS – This is a card sequence number. Column two contains the high-order 12 bits of the number and column three contains the low-order 6 bits (rows 12-3).
Column 4:	ENTRY COUNT – This is a 6-bit count of the number (rows 4-9 only) of disk files defined on this card.
Column 5:	CHECK SUM – This is the sum of all columns except 5, 76-80.
Columns 6-10:	Blank.
Columns 11-12:	MAXIMUM NUMBER OF WORDS – This is the number of 60-bit words in the longest logical record. Column 11 (rows 4-9) contains the high-order 6 bits and column 12 the low-order 12 bits.
Column 13:	LOGICAL DISK NUMBER – This is a 12-bit number which identifies the logical number of the disk file.
Columns 14-75:	OTHER ENTRIES.
Columns 76-80:	DECK ID.

Central Memory Program Name Card

These cards identify the central processor program, define the length of the program, and designate the program entry point.

Column 1:	CARD ID – 12-bit indentification code (0305).
Columns 2-3:	LOAD ADDRESS – This is a card sequence number. Column two contains the high-order 12 bits of the number and column three contains the low-order 6 bits (rows 12-3).
Column 4:	ENTRY COUNT – This is a 6-bit count (row 4-9) of the number of programs named by the card.
Column 5:	CHECK SUM – This is the sum of all columns except 5, 76-80.
Columns 6-10:	Blank.
Columns 11-14:	PROGRAM NAME – This is the name which identifies the program and consists of one to eight 6-bit BCD character codes starting in column 11.
Columns 15-16:	PROGRAM ENTRY POINT – This is the relative address of the entry point to the program. Columns 15 contains (rows 4-9) the high-order 6 bits of the address and column 16 contains the low-order 12 bits.
Columns 17-75:	Blank.
Columns 76-80:	Deck ID.

Central Memory Subroutine Name Card

These cards identify a central processor subroutine, define the length of the subroutine, and designate the subroutine entry point.

Column 1:	CARD ID – 12-bit identification code (0205).
Columns 2-3:	LOAD ADDRESS – This is a card sequence number. Column two contains the high-order 12 bits of the number and column three contains the low-order 6 bits (rows 12-3).
Column 4:	ENTRY COUNT – This is a 6-bit count (rows 4-9) of the number of subroutines named by the card.
Column 5:	CHECK SUM – This is the sum of all columns except 5, 76-80.
Columns 6-10:	Blank.
Columns 11-14:	SUBROUTINE NAME – This is the name which identifies the subroutine; it consists of from one to eight 6-bit BCD characters starting in column 11.
Columns 15-16:	SUBROUTINE ENTRY POINT – This is the relative address of the entry point to the subroutine. Column 15 contains (row 4-9) the high-order 6 bits of the address and column 16 contains the low-order 12 bits.
Columns 17-75:	OTHER ENTRIES.
Columns 76-80:	DECK ID.

Subroutines References

This card contains a list of all routines called by this routine but not assembled with it.

Column 1:	CARD ID – 12-bit identification code (0105).

Columns 2-3:	LOAD ADDRESS – This is a card sequence number. Column two contains the high-order 12-bits of the number and column three contains the low-order 6 bits (rows 12-3).
Column 4:	ENTRY COUNT – This is a 6-bit count (rows 4-9) of the number of subroutines named by the card.
Column 5:	CHECK SUM–This is the sum of all columns except 5, 76-80.
Columns 6-10:	Blank.
Columns 11-14:	NAME OF SUBROUTINE – This is the name of the subroutine called by the program but not assembled with it and consists of one to eight 6-bit BCD characters starting in column 11.
Columns 15-16:	LOCATION OF LAST REFERENCE – This is the relative address of the last reference to the subroutine referenced. Column 15 contains (rows 4-9) the high-order 6 bits of the address and column 16 contains the low-order 12 bits.
Columns 17-75:	OTHER ENTRIES.
Columns 76-80:	DECK ID.

Central Processor Program Text Cards

Central processor program text cards contain thirteen central processor instructions each, along with relocation information for each instruction.

Column 1:	CARD ID – 12-bit identification code (2005 or 6005).
Columns 2-3:	LOAD ADDRESS – This is the number of locations from the start of the routine to the first instruction word. Column two contains the high-order 12-bits of the number and column three contains (rows 12-3) the low-order 6 bits.
Column 4:	WORD COUNT – This is the number (from 1 to 13) of central memory words contained in the card.
Column 5:	CHECK SUM – This is the sum of all columns except 5, 76-80.
Columns 6-10:	RELOCATION OF DIRECTORY – This field has one 4-bit binary code for each of the thirteen central processor instructions in the card. The code for the first word is in column 6, rows 12-1, the second is in column 6, rows 2-5, etc.
Columns 11-15:	WORD ONE – This is the first central processor instruction in the card.
Columns 16-75:	OTHER INSTRUCTIONS.
Columns 76-80:	DECK ID.

PERIPHERAL PROCESSOR CARDS

Peripheral Processor Program Name Card

This card is used to identify the peripheral processor program and tells the Job Loader how much central memory space is to be reserved for the program.

Column 1:	CARD ID – 12-bit identification code (0307).
Columns 2-3:	Blank.
Column 4:	ENTRY COUNT – This is a 6-bit count (rows 4-9) of the entries in the card and is always 01 for peripheral processor name cards.
Column 5:	CHECK SUM – This is the sum of all columns except 5, 76-80.
Columns 6-7:	Blank.
Columns 8-9:	AMOUNT OF SPACE – This is an 18-bit number identifying the number of central memory words reserved by the peripheral processor program. Column 8 contains (rows 4-9) the high-order 6 bits of the address and column 9 contains the low-order 12 bits.
Column 10:	Blank.
Columns 11-14:	PROGRAM NAME – This consists of up to eight 6-bit BCD characters starting in column 11.
Columns 15-75:	Blank.
Columns 76-80:	DECK ID.

Disk Space Requirement Card

The disk space requirement card defines the disk files used in the peripheral program.

Column 1:	CARD ID – 12-bit identification code (0407).
Columns 2-3:	Blank.
Column 4:	ENTRY COUNT – This is a 6-bit count (rows 4-9) of the number of disk files defined by the card.
Column 5:	CHECK SUM – This is the sum of all columns except 5, 76-80.
Column 6:	LOGICAL DISK NUMBER–This is the logical number of the disk unit.
Columns 7-8:	MAXIMUM NUMBER OF CM WORDS – This is the number of 60-bit words in the longest logical record. Column 7 (rows 4-9) contains the high-order 6 bits and column 8 contains the low-order 12 bits.
Columns 9-10:	LOCATION OF LOGICAL RECORD TABLE – This is the relative address of the logical record table in central memory. The address is relative to the start of the space reserved by columns 8-9 of the program name card. Column 9 contains (rows 4-9) the high-order 6 bits and column 10 contains the low-order 12 bits of the address.
Columns 11-12:	MAXIMUM NUMBER OF LOGICAL RECORDS–This is the maximum number of logical records in the first disk file defined on the card. Column 11 contains (rows 4-9) the high-order 6 bits and column 10 contains the low-order 12 bits of the address.
Columns 13-75:	OTHER ENTRIES.
Columns 76-80:	DECK ID.

Peripheral Processor Program Text Card

Peripheral processor program text cards contain the instructions for the peripheral processor program and the relocation codes which tell how the addresses of instructions are to be relocated.

Column 1:	CARD ID – 12-bit identification code (2007 or 6007).
Column 2:	LOAD ADDRESS – This is the number of locations from the start of the routine to the first instruction word.
Column 3-4:	WORD COUNT – This is the 6-bit count (rows 4-9) of the number of 12-bit words starting in column 16.
Column 5:	CHECK SUM – This is the sum of all columns except 5, 76-80.
Columns 6-15:	RELOCATION DIRECTORY – This field contains one 2-bit relocation code for each instruction word in the card. Rows 12-11 of column 6 contain the code for the first instruction word, rows 0-1 of column 6 contain the code for the second instruction word, and so on.
Columns 16-75:	INSTRUCTION WORDS – Each 12-bit column contains one peripheral processor instruction.
Columns 76-80:	DECK ID.

Peripheral Processor Subroutine Overlay Name Card

This card defines the part of the peripheral processor program to be overlaid.

Column 1:	CARD ID – 12-bit identification code (0207).
Columns 2-3:	Blank.
Column 4:	ENTRY COUNT – This is a 6-bit count (rows 4-9) of entries but is always 01 for this card.
Column 5:	CHECK SUM – This is the sum of all columns except 5, 76-80.
Columns 6-10:	Blank.
Columns 11-14:	NAME OF OVERLAY – This is the name of the program segment to be overlaid and consists of from one to eight 6-bit BCD characters.
Columns 15-75:	Blank.

Columns 76-80: DECK ID.

Termination Card

This card identifies the end of the central processor and associated peripheral processor programs in a job.

Column 1:	CARD ID — 12-bit identification code (0015).
Columns 2-75:	Blank.
Columns 76-80:	DECK ID.

Pub. No. 60101800A